UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

ANÁLISE COMPARATIVA DE ALGUNS ALGORITMOS NA SOLUÇÃO DO PROBLEMA DA MOCHILA

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA

PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

IVAN LUDGERO IVANQUI

FLORIANÓPOLIS

SANTA CATARINA - BRASIL

1986

ANÁLISE COMPARATIVA DE ALGUNS ALGORITMOS
NA SOLUÇÃO DO PROBLEMA DA MOCHILA

IVAN LUDGERO IVANQUI

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE "MESTRE EM ENGENHARIA"

ESPECIALIDADE ENGENHARIA DE PRODUÇÃO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO.

PROF. PAULO RENÉCIO NASCIMENTO, M.Sc.
ORIENTADOR

PROF. ROBERT WAYNE SAMOHYL, Ph.D.
COORDENADOR

BANCA EXAMINADORA:

Paulo Nascimento
PROF. PAULO RENÉCIO NASCIMENTO, M.Sc.
PRESIDENTE

PROF. LUIZ GONZAGA DE SOUZA FONSECA, D.Sc.

PROF. RAUL VALENTIM DA SILVA, M.Sc.

a Nereide, Marcos e Ivana

AGRADECIMENTOS

Ao Prof. Paulo Renécio Nascimento, pela excelente orientação e pelo estímulo à execução deste trabalho.

À Universidade Estadual de Maringá por ter-me proporcionado condições para desenvolver este trabalho.

À Universidade Federal de Santa Catarina pelo apoio técnico.

A todas as pessoas que, direta ou indiretamente, contribuíram para a realização deste trabalho.

RESUMO

O objetivo do presente trabalho é fazer uma análise comparativa de alguns algoritmos na solução do problema da mochila.

Inicialmente, situam-se os algoritmos a serem trabalhados dentro de suas respectivas famílias.

Segue-se com a apresentação de um algoritmo de busca vertical direta para o problema da mochila, proposto por ZOLTNERS.

Posteriormente, situa-se a técnica lexicográfica dentre os métodos de solução de programação inteira. Apresentam-se alguns algoritmos que usam esta técnica na solução de problemas de programação linear inteira e uma variante destes algoritmos para a solução específica do problema da mochila, denominado LEXMOCH.

Finalmente, é apresentada uma análise comparativa quanto ao desempenho computacional destes algoritmos.

ABSTRACT

The objective of the present work is to make a comparative analysis of some algorithms in solving the knapsack problem.

Firstly, the algorithms to be worked are situated inside their respective family.

Then a ZOLTNERS' Direct Descent Binary knapsack algorithm is presented.

Afterwards, a lexicographic technique among the solution methods of integer programming is demonstrated. Some algorithms are presented that use this technique in solving integer linear programming problems and a variation of these algorithms is developed for the solution of a especific knapsack problem, called LEXMOCH.

Finally, a comparative analysis is made refering to the computational efficiency of these algorithms.

SUMÁRIO

CAPÍTULO I

1. O PROBLEMA DA MOCHILA

1.1. Introdução

O nome "problema da mochila" deriva de uma situação hipotética. Considere um andarilho que está carregando uma mochila em sua viagem. Ele deve ocupar sua mochila escolhendo, entre muitas coisas, aquelas que têm determinada importância pessoal e peso. Certamente, ele escolherá carregar um máximo de objetos pessoais necessários com o menor peso (veja HU[1]).

1.2. Formulação do Modelo Matemático

O problema da mochila situa-se entre os problemas de programação linear, inteira e binária. É definido por:

$$\max Z = C.X$$

$$\text{sujeito a} \quad A.X \leq b$$

onde

C - é um vetor n-dimensional de benefícios

A - é um vetor n-dimensional de recursos

X - é um vetor n-dimensional de variáveis de decisão

b - é o recursos total disponível

Z - é o valor total dos benefícios ou valor da função objetivo

As componentes do vetor X vão assumir os valores 0 ou 1, com o seguinte significado.

X_i = 1 - deve ser alocado o i-ésimo recurso

X_i = 0 - não deve ser alocado o i-ésimo recurso

As componentes c_i, a_i e a constante b são inteiras e positivas.

1.3. Aplicações do problema da Mochila

Segundo SALKIN e KLUYVER[2]:

"Embora o problema da Mochila represente muitas situações práticas como: seleção de projetos, controle de investimento, carregamento de carga, etc., poucas aplicações diretas são encontradas na literatura".

"LORIE e SAVAGE[3] descrevem, em seu artigo, aplicações em investimento de capital".

"WEINGARTNER[4,5,6] discutiu as aplicações de Lorie-Savage e estendeu o modelo para os casos onde os projetos não são independentes. Em particular, o caso onde a função objetivo é quadrática foi tratado por UNGER[7], RADHAKRISHNAN[8] e por MAO e WALLINGFORD[9]. O uso da teoria dual de BALAS[10,11,12] na programação inteira, para dar interpretação econômicas e propriedades da solução ótima (para modelos lineares e não lineares) é apresentado por RADHAKRISHNAN[8]. Outros artigos relativos à programação matemática aplicada ao investimento de capital incluem os de BAUMOL e QUANDT[13], BYRNE, CHARNES, COOPER e KORTANEK[14], NASLUND[15], e WOOLSEY[16] ".

"Um recente trabalho de GLOVER e KLINGMAN[17] apresenta uma aplicação para o problema na seleção de jornais numa livraria. Outros artigos anteriores que descrevem este

problema foram escritos por KRAFT e HILL[18,19]".

"Relacionado ao problema da mochila está o problema de "Cutting-stock" relatado por GILMORE e GOMORY[20,21,22] . Ele é descrito como um problema de programação linear onde uma coluna representa um certo padrão de corte".

"Uma aplicação recente do problema da Mochila está no uso da teoria dos grupos em programação inteira. Em particular, GOMORY e JOHNSON[23] apresentaram a maneira de relaxar as condições de viabilidade em certas variáveis e assim qualquer problema inteiro pode ser definido como problema inteiro num grupo abeliano. Outrossim, para este problema, os algoritmos, do tipo de programação dinâmica (ver HU[1]), tratam a relação de grupo como uma simples restrição e assim são essencialmente adaptados para resolver o problema da mochila".

Outra aplicação foi a apresentada por MENDES[24] na solução de um problema de Alocação Multi-dimensional.

1.4. Famílias de algoritmos existentes

Segundo SALKIN e KLUYVER[2], as várias técnicas desenvolvidas para a solução do problema da Mochila, situam-se entre as quatro famílias:

a) algoritmos de enumeração implícita;

b) algoritmos de busca em grafos;

c) algoritmos heurísticos e métodos lagrangeanos;

d) algoritmos de programação dinâmica.

Dentre estes métodos, far-se-á uma breve explanação dos ítens a e b por serem as famílias que caracterizam os al-

goritmos que serão desenvolvidos neste trabalho.

1.4.1. Algoritmos de enumeração implícita

Um algoritmo é dito de enumeração implícita, quando assegura a determinação da solução ótima, sem que seja necessário examinar todo o conjunto de soluções.

Estudos deste processo foram realizados por KOLESAR[25] (1967) e GREENBERG E HEGERICH[26] (1970) utilizando a técnica de "Branch and Bound". Logo após GUIGNARD [27] (1972) apresenta um algoritmo heurístico baseado no algoritmo aditivo de Balas.

1.4.2. Algoritmos de busca em grafos

É um procedimento sistemático de geração de sub-grafos, visando encontrar uma solução para o caminho de mínimo custo.

Aplicações deste processo para o problema da mochila foram feitas por: SHAPIRO[28] (1968), DREYFUS[29] (1969), SHAPIRO e WAGNER[30] (1966), HU[1] (1969), NEMHAUSER e GARFINKEL[31] (1972) e outros.

1.5. Classificação dos algoritmos apresentados

O algoritmo apresentado no capítulo 2, proposto por ZOLTNERS[32] e denominado "A Direct Descent Binary Knapsack Algorithm", pode ser caracterizado como um procedimento de busca em grafos. Outrossim, é importante ressaltar que também é possível classificá-lo na família dos algoritmos de enumeração

implícita, pois assegura a determinação da solução ótima sem examinar todas as soluções possíveis.

O algoritmo de ALVAREZ[33], modificado para o problema da Mochila, apresentado na seção 3.3, é classificado como um método de enumeração implícita, usando o processo de ordenação lexicográfica para enumerar as soluções. Por outro lado, pode ser considerado um algoritmo de busca em grafos, pois, à medida que se aprofunda na busca, vai gerando uma arborescência (sub-grafo) visando encontrar uma solução para o caminho de mínimo custo.

CAPÍTULO II

2. ALGORITMO DE BUSCA VERTICAL DIRETA PARA O PROBLEMA DA MOCHILA

2.1. Descrição do algoritmo de ZOLTNERS

O algoritmo apresentado por ZOLTNERS[32], pressupõe em sua fase inicial uma ordenação decrescente de índices segundo o valor da relação c_j/a_j e foi desenvolvido em cinco passos distintos:

1 - Teste de viabilidade

2 - Alocação

3 - Redução

4 - Teste de retorno

5 - Retorno

Far-se-á, em seguida, uma descrição detalhada para cada passo.

2.1.1. Teste de viabilidade

Para este passo, o artigo prevê o seguinte procedimento:

"Um retorno (backtrack) é possível em qualquer ponto da busca, se puder ser mostrado que qualquer busca posterior não produzirá uma solução melhor que a obtida até o momento. A posição da busca será sempre caracterizada pelo índice da variável que está sendo examinada e pelo conjunto de variáveis

com valor atual igual a um (1), dentre as p-1 variáveis iniciais. Esta pesquisa é realizada na ordem decrescente de índices de modo que variáveis já pesquisadas não são mais consideradas.

Considerando que o nível da busca é k, tem-se:

$$Z_k = \Sigma c_j \quad \text{para} \quad j \in N_k^1$$

$$S = b - \Sigma a_j \quad \text{para} \quad j \in N_k^1$$

$$\hat{Z}_k = Z_k + C_{k+1} \times S/a_{k+1}$$

onde:

N_k^1 - é o conjunto das variáveis que assumem o valor um (1) no nível k da busca.

Z_k - é o valor da solução obtida até a posição k.

b - é o recurso total disponível.

$\hat{Z}_k$ - é uma solução estimada para a posição k+1 em função dos recursos disponíveis.

S - é a quantidade de recursos ainda disponíveis para utilização.

O valor $\hat{Z}_k$ quando comparado com o da melhor solução inteira obtida (Z^*), determinará se a busca deve ser aprofundada nesta direção. Em caso afirmativo, o algoritmo desvia o processo para o passo alocação; caso contrário, provocará o desvio para o passo onde se faz o teste de retorno.

2.1.2. Alocação

Para todo ramo k da busca, o processo de alocação se realiza da seguinte forma:

a) Verificar se é possível alocar o recurso referente à posição k;

b) se possível, tornar a variável de decisão unitária, obtendo na alocação o benefício associado e atualizar os recursos disponíveis. Caso contrário, a variável de decisão assume o valor zero, implicando na não alocação do recurso. Em ambos os casos, retorna ao passo anterior (teste de viabilidade);

c) a saída deste processo para o passo seguinte só ocorrerá quando se estiver testando o n-ésimo recurso, ou se já se tiver ocupado todos os recursos disponíveis (S=0), com isto obtem-se uma nova solução.

2.1.3. Redução

Será realizado este passo toda vez que uma nova solução tiver sido determinada.

O desenvolvimento deste será através das etapas:

a) Definir p, índice que corresponde à posição da variável fracionária, na solução do problema contínuo.

b) Calcular $\bar{c}_j$, fator relativo de custos, para todas as variáveis do problema, usando a relação:

$$\bar{c}_j = c_j - (c_p/a_p)a_j \qquad 1 \leq j \leq n,$$

(p é o índice que representa a posição da variável com valor fracionário no problema contínuo).

c) Calcular $\bar{Z}_k^h$, limite superior do valor da função objetivo enquanto $x_k = h$, nas soluções viáveis, através de:

$$\text{Se } h = 0 \rightarrow \bar{Z}_k^0 = \bar{Z} - \bar{C}_k \text{ para } 1 \leqq k \leqq p - 1$$

$$\text{Se } h = 1 \rightarrow \bar{Z}_k^1 = \bar{Z} + \bar{C}_k \text{ para } p + 1 \leqq k \leqq n$$

Caso $\bar{Z}_k^h \geqq Z^*$, ou seja, a melhor solução até a presente iteração for menor que a estimativa do limite superior da função objetivo para X_k=h, então, necessariamente, $X_k = 1-h$ para uma nova solução inteira melhor que a atual. Esta mudança, no valor de X_k, determinará novos sub-problemas a serem testados.

2.1.4. Teste de Retorno

Inicialmente, as (p-1) variáveis iniciais assumem o valor um (1). À medida que o procedimento continua e retornos são executados, algumas dessas variáveis tornam-se iguais a zero. Quando a primeira variável é fixada em 0, ela não muda para o resto da busca, o mesmo ocorrendo para as variáveis subseqüentes. Conseqüentemente, a busca reduz o conjunto N_k^1 à medida que se aprofunda a análise naquela direção. Observe-se que o algoritmo sempre retorna para uma variável previamente fixada em um (1), uma vez que, quando a busca é terminada, o conjunto das variáveis fixadas em um (1) é vazio.

2.1.5. Retorno (backtracking)

É possível realizar um retorno, quando:

a) uma nova solução tenha sido encontrada;

b) a pesquisa segundo a direção tomada não resulte numa solução melhor.

Existem duas maneiras distintas para se realizar o retorno (backtracking), que dependem exclusivamente da variável de decisão ter assumido o valor zero ou um, ao término da busca naquela direção:

a) se a busca terminar segundo um valor zero, deve-se retornar a uma variável de decisão anterior, com o valor um, e fixá-la em zero, conforme mostram as figuras 1 e 2,

b) se a busca terminar segundo um valor um (1) deve-se retornar à primeira variável unitária, anterior a ramos zeros intercalados, e fixá-la em zero. (Vide figuras 3 e 4).

Retorno realizado quando a busca termina segundo um ramo zero:

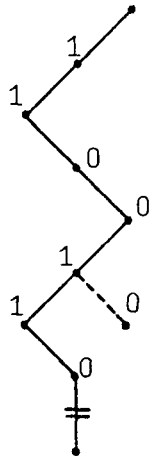


FIGURA 1

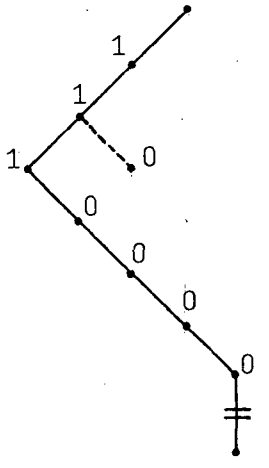


FIGURA 2

Retorno realizado quando a busca termina segundo um ramo um (1):

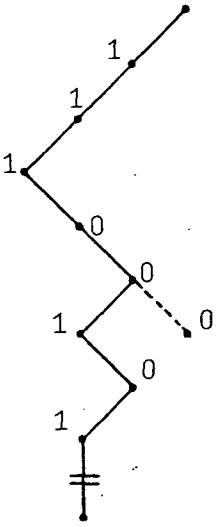


FIGURA 3

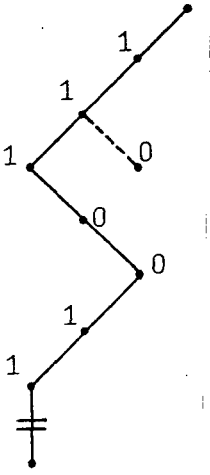


FIGURA 4

O processo iterativo envolvendo as etapas já apresentadas está representado no fluxograma da figura 5.

2.2. Dificuldades encontradas na implementação do algoritmo de ZOLTNERS.

Apesar de terem sido objeto de muito estudo, alguns ítens do algoritmo de Busca Vertical Direta apresentados no artigo de ZOLTNERS[32] geraram sérios problemas de interpretação, devido à maneira dúbia com que foram apresentados. Assim, a versão final encontrada foi denominada de ALGORITMO DE ZOLTNERS MODIFICADO.

O problema primordial foi a definição do valor do índice p, que representa a posição da variável fracionária na solução do problema contínuo da mochila. A idéia apresentada para o cálculo de p na solução inteira inicial, como sendo a po-

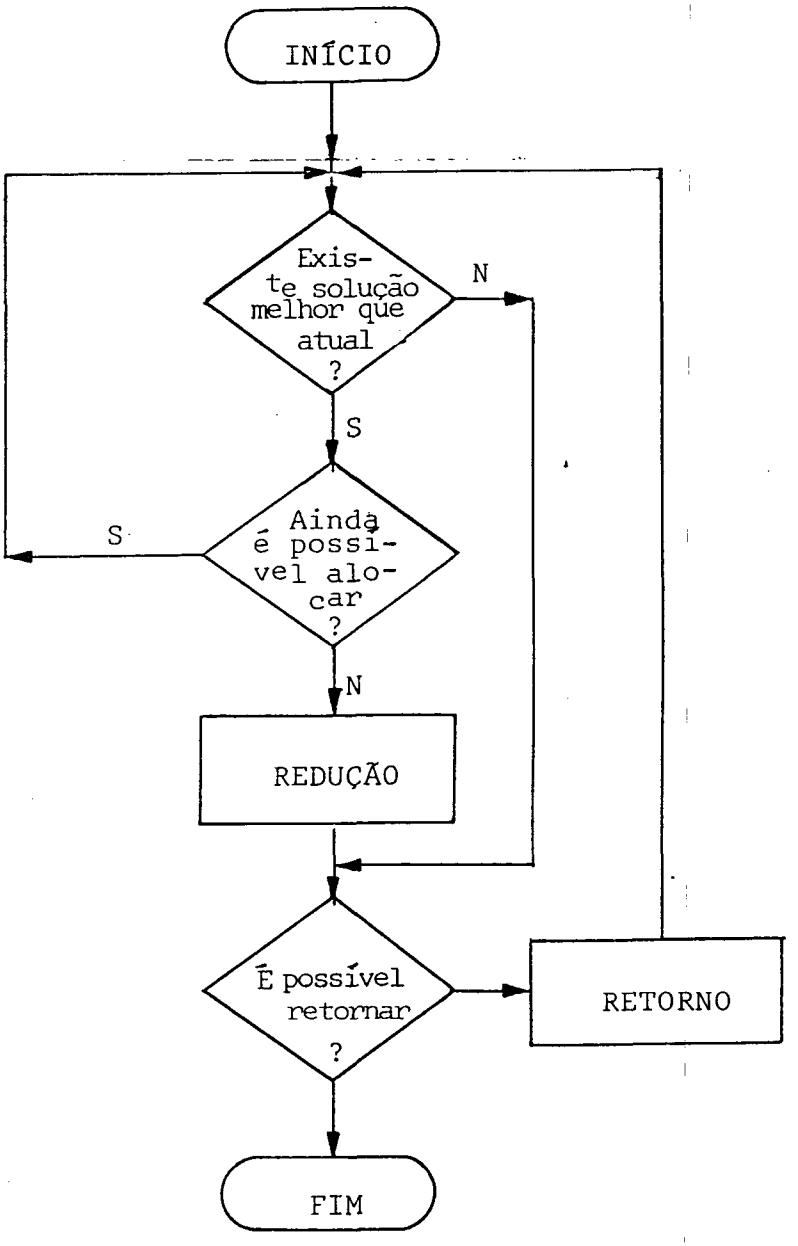


FIGURA 5 - Fluxograma da iteração das etapas propostas no algoritmo de Zoltners.

sição do primeiro elemento zerado, estava clara; o que não ocorria para as próximas soluções encontradas. Deste modo foram levantadas várias hipóteses, a saber:

1) fazer p variar de forma decrescente desde seu valor inicial até um (1);

2) fazer p constante e igual ao valor encontrado na solução inicial;

3) fazer p variar de forma crescente desde seu valor inicial até a última posição;

4) fazer p variar de seu valor inicial até um (1), assumindo apenas os valores correspondentes às variáveis que devem ser pesquisadas para cada nova solução encontrada.

Dentre as tentativas expostas acima, a que deu mais resultado foi a de número 4, apesar de não ter apresentado um índice de redução maior que o das outras. Esta tentativa transformou o algoritmo em admissível, o que não ocorria com as demais.

Juntamente com as proposições acima, foi modificado, para efeito de estudo, o teste de viabilidade, anteriormente não realizado para a obtenção da solução inicial. Para as demais, aplicava-se apenas para a variável fixada no retorno. Este fato motivou as seguintes mudanças no passo onde se realiza o processo de Alocação:

1) alocar os recursos segundo a ordem decrescente de índices, definidos pelos valores correspondentes de $\bar{Z}_k^h$, a partir da posição indicada no retorno;

2) alocar os recursos possíveis, em ordem crescente de índices, a partir da posição indicada no retorno, até a última

variável.

Estas duas hipóteses de alocação foram combinadas com as quatro iniciais de variação do valor de p. Dos resultados obtidos, a primeira hipótese foi logo abandonada em virtude de transformar o algoritmo em apenas completo. A segunda alternativa, juntamente com a quarta, para o valor de p, se mostrava mais promissora em virtude dos resultados obtidos na solução dos problemas.

Outra dificuldade ocorreu no passo onde se realiza a Redução e se definem as variáveis a serem pesquisadas para cada nova solução obtida. Este passo, ao se interpretar erroneamente as informações contidas no artigo, retirava a admissibilidade do algoritmo. Isto em virtude do cálculo de Z_k^h, limite superior do valor da função objetivo enquanto X_k=h, que é definido por:

$$\text{Se } h = 0 \rightarrow Z_k^0 = \bar{Z} - \bar{C}_k \quad \text{para } 1 \leq k \leq p - 1$$

$$\text{Se } h = 1 \rightarrow Z_k^1 = \bar{Z} + \bar{C}_k \quad \text{para } p + 1 \leq k \leq n$$

Ser calculado tomando-se para $\bar{Z}$ a solução encontrada naquela direção, em lugar de se estimar este valor em função do que ainda resta por alocar e a posição que definiu esta direção.

A versão do algoritmo já incluindo todas as mudanças apresentadas resolveu os 750 problemas propostos encontrando a solução ótima.

Somente após ter-se uma versão admissível, é que foi realizado um estudo mais detalhado do teste de viabilidade. Ficou claro na solução de alguns problemas, que este pro-

cesso reduz o número de testes de alocações.

O fluxograma do algoritmo de ZOLTNERS[32] é apresentado na figura 6.

2.3. Algoritmo de ZOLTNERS modificado.

2.3.1. Alterações realizadas no algoritmo proposto por ZOLTNERS.

As modificações necessárias para que o algoritmo tivesse o desenvolvimento segundo o apresentado no artigo são as seguintes:

1) Cálculo do índice p

A idéia apresentada no artigo, para este cálculo, na solução inicial, era clara; o que não ocorria para as demais soluções, conforme já exposto no item 2.2.

A solução encontrada foi:

"Fazer p variar de seu valor inicial até um (1), assumindo apenas os valores correspondentes às variáveis que devem ser pesquisadas, para cada nova solução encontrada.

As variáveis que devem ser pesquisadas serão as definidas pelo teste $\bar{Z}_k^h \geq Z^*$ enquanto $X_k = h$ na Redução. Como pode-se observar nos exemplos a seguir, as variáveis que podem ser pesquisadas serão as representadas sem o (*) nas respectivas soluções.

2) Cálculo de $\bar{Z}$.

É dado pela relação:

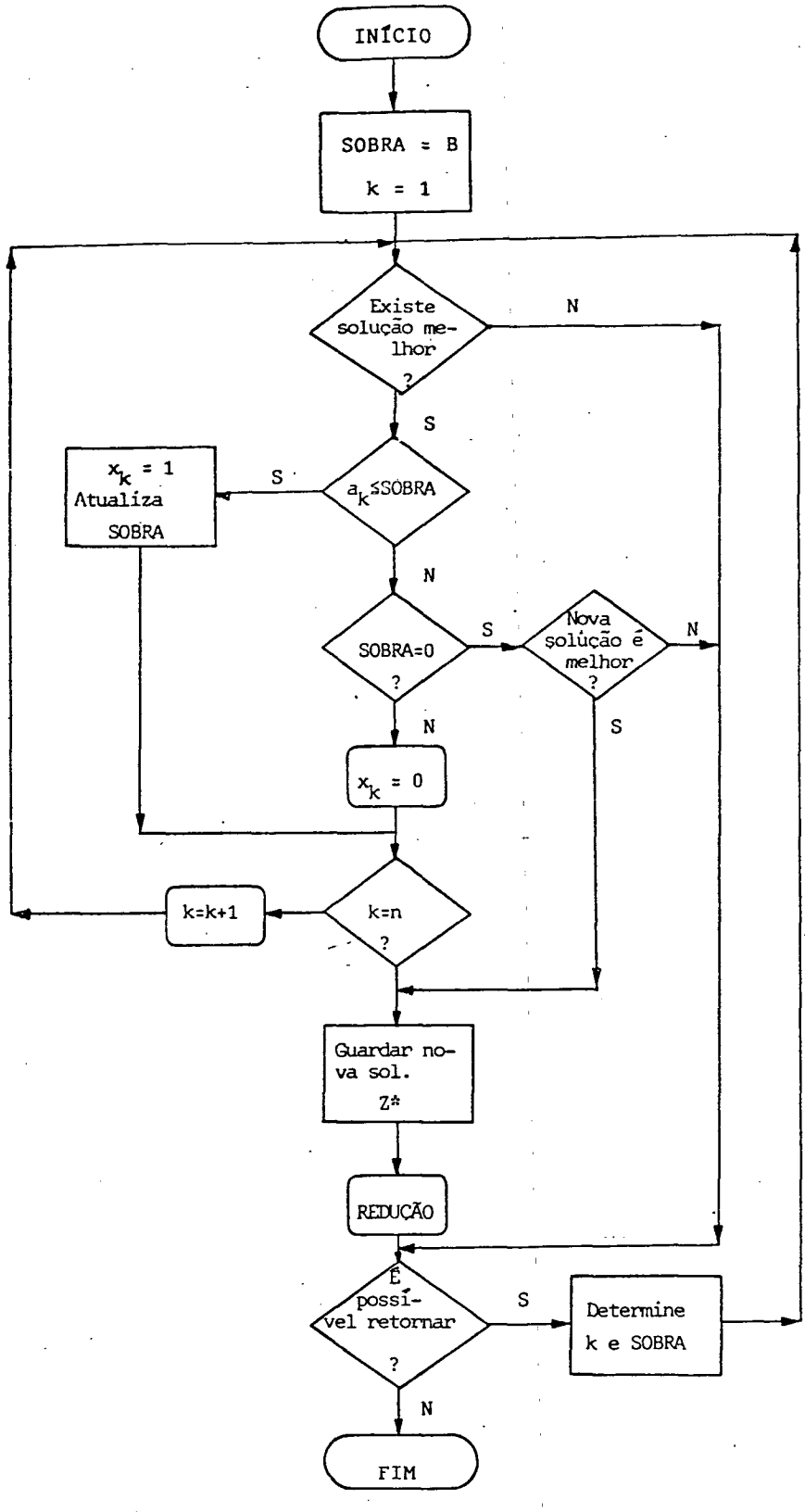


FIGURA 6 - Fluxograma do algoritmo de Zoltners.

$$\bar{Z} = Z + C_p S/a_p$$

Onde:

Z - é a solução inteira encontrada até o nível k da busca,

C_p - é a p - ésima componente do vetor benefício,

a_p - é a p - ésima componente do vetor recurso,

S - é a quantidade de recursos disponíveis para utilização.

É importante observar que:

a) os resultados do item 1, estão relacionados ao deste, pois usa-se para o cálculo de $\bar{Z}$ a p-ésima componente dos vetores de benefícios e recursos;

b) o cálculo de $\bar{Z}$ está relacionado ao de $\bar{Z}_k^h$, que define as variáveis que devem ser pesquisadas;

c) o algoritmo, após estas alterações, transformou-se em admissível.

O fluxograma do algoritmo de ZOLTNERS modificado é apresentado na figura 7.

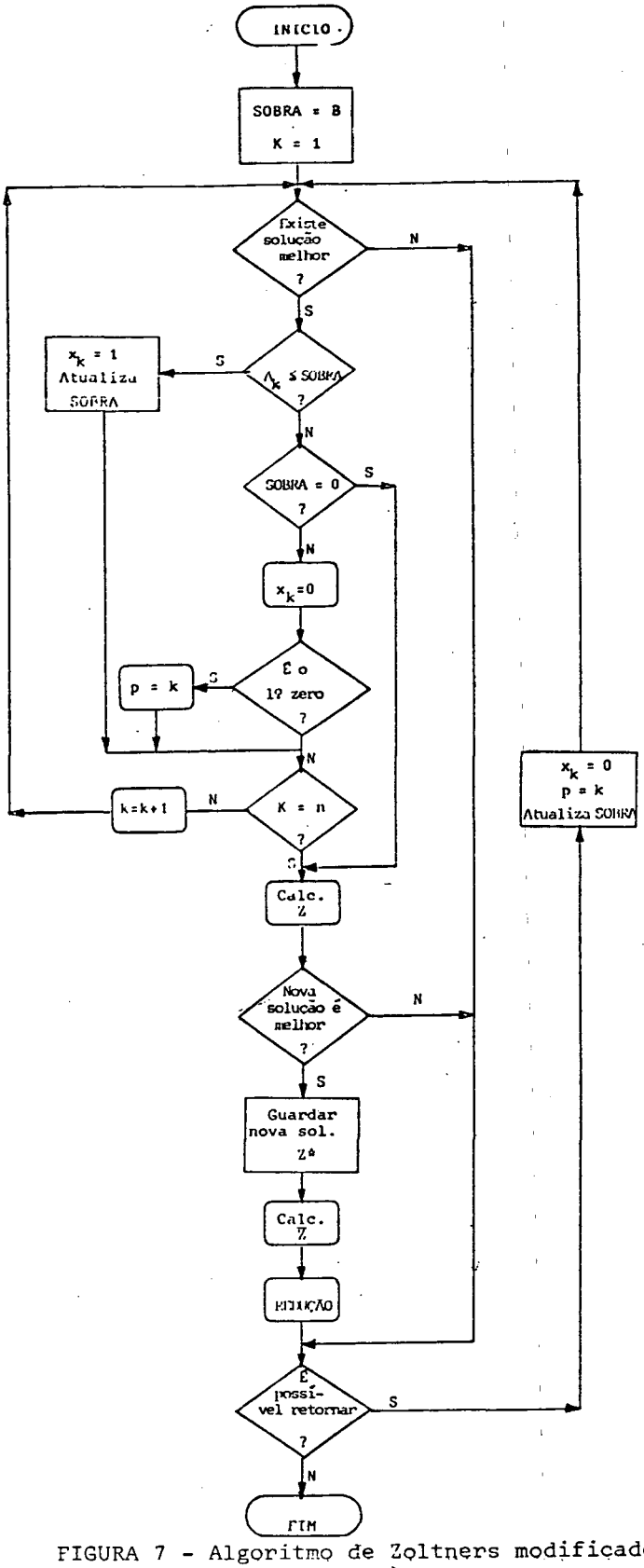


FIGURA 7 - Algoritmo de Zoltners modificado.

2.3.2. Exemplos:

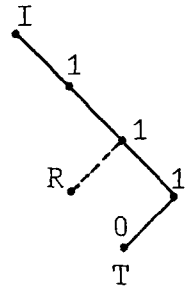
A seguir são mostrados graficamente, quais as soluções testadas em cada problema e o valor do p respectivo.

Exemplo 1:

$$\text{Max } Z = 3x_1 + 4x_2 + 5x_3 + 6x_4$$

$$\text{S.A.} \quad x_1 + 2x_2 + 3x_3 + 4x_4 \leq 8$$

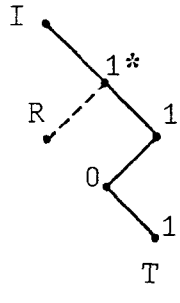
ITERAÇÃO 1



I = Início da busca
T = Término da busca
R = Retorno

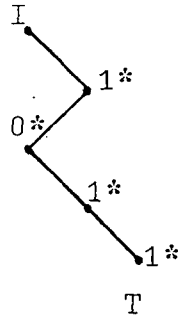
P = 4
Z = 12
X = (1, 1, 1, 0)

ITERAÇÃO 2



P = 3
Z = 13
X = (1, 1, 0, 1)

ITERAÇÃO 3



P = 2
Z = 14
X = (1, 0, 1, 1)

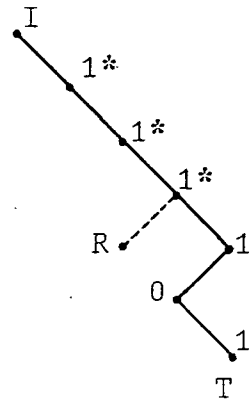
Solução encontrada $Z^* = 14$ $X^* = (1, 0, 1, 1)$.

Exemplo 2.: Max $Z = 8x_1 + 20x_2 + 8x_3 + 2x_4 + 10x_5 + 2x_6$

S.a. $x_1 + 4x_2 \quad 4x_3 \quad 3x_4 \quad 20x_5 \quad 6x_6 \leq 28$

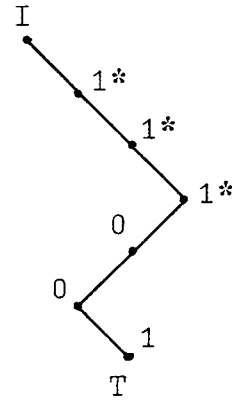
I = Início da busca
T = Término da busca
R = Retorno

ITERAÇÃO 1



P = 5
Z = 40
X = (1, 1, 1, 1, 0, 1)

ITERAÇÃO 2



P = 4
Z = 38
X = (1, 1, 1, 0, 0, 1)

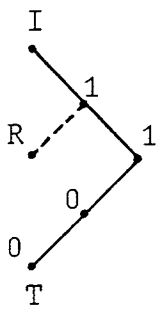
Obs.: A busca termina, pois os próximos retornos ocorreriam em variáveis com (*). A solução é $Z^* = 40$ e $X^* = (1, 1, 1, 1, 0, 1)$.

Exemplo 3.: max $Z = 7x_1 + 20x_2 + 8x_3 + 2x_4$

S.a. $x_1 + 3x_2 + 4x_3 + 4x_4 \leq 7$

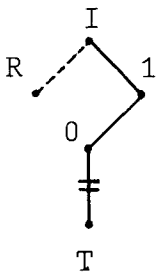
I = Início da busca
T = Término da busca
R = Retorno

ITERAÇÃO 1

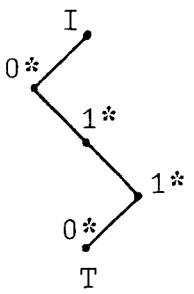


P = 3
Z = 27
X = (1, 1, 0, 0)

ITERAÇÃO 2



P = 2



P = 1
Z = 28
X = (0, 1, 1, 0)

Como se pode observar na iteração 2, foi realizado um Retorno (Backtrack), pois as buscas resultantes da segunda variável, não são superiores à melhor solução obtida até o momento. A segunda figura da iteração 2 nos mostra a nova solução encontrada após o Retorno.

Solução encontrada:

$Z^* = 28$ $X^* = (0, 1, 1, 0)$

CAPÍTULO III

3. ALGORITMOS LEXICOGRÁFICOS

3.1. Introdução

Dentre os métodos de enumeração implícita encontram-se aqueles baseados em técnicas lexicográficas, sendo que o algoritmo básico utilizando estas técnicas, foi inicialmente desenvolvido por LAWLER e BELL[34] (1966), seguido dos apresentados por WALLINGFORD[35] (1969), DRAGAN[36] (1968 - 1969), RÖDDER[37] (1972) e ALVAREZ[33] (1979).

Os algoritmos acima descritos são utilizados para resolver problemas de programação linear inteira. Como o problema da mochila é um subproblema destes, desenvolveu-se neste trabalho um algoritmo usando a técnica lexicográfica para a solução deste tipo particular de problema.

Dado que o presente trabalho desenvolve-se baseado no algoritmo apresentado por ALVAREZ[33], apresenta-se a seguir a técnica de RÖDDER[37] e a de ALVAREZ[33], que dela se originou.

3.2. Conceitos básicos

Para compreensão do assunto tratado neste capítulo, é necessário o conhecimento prévio de alguns conceitos:

- ORDENAÇÃO LEXICOGRÁFICA.

O vetor Y é lexicograficamente maior que o vetor 0 (ze-

ro) se e somente se a primeira componente não nula do vetor Y é positiva.

Notação:

$$Y \succ 0$$

O vetor Y é lexicograficamente maior que o vetor X se e somente se a diferença Y - X for lexicograficamente maior do que o vetor 0 (zero).

Notação:

$$Y \succ X \text{ se e somente se } Y-X \succ 0$$

3.3. Algoritmo Lexicográfico de Rödder - "LEXS"

3.3.1. Formulação do Modelo Matemático.

Para a utlização do algoritmo lexicográfico (LEXS) na solução de problemas de programação linear inteira, eles deverão estar na forma:

$$\text{Max } Z = \Sigma a_{0j} x_j \qquad 1 \leq j \leq n$$

$$\text{Tal que } \sum_j a_{ij} x_j - b_i \geq 0 \qquad 1 \leq i \leq m \quad 1 \leq j \leq n$$

onde:

$$x_j \in D_j = \{0, 1, \ldots, I_j\} \quad 1 \leq j \leq n$$

a_{ij} - pertence ao conjunto dos números inteiros

I_j - limite superior da variável X_j

D_j - conjunto de valores que a variável X_j pode assumir

m - número de restrições

n - número de variáveis

3.3.2. Modificações introduzidas por Rödder no algoritmo de Lawler e Bell.

O trabalho de RÖDDER[37] foi desenvolvido com base no trabalho de LAWLER e BELL[34], apresentando as seguintes variações:

- Condensa vários saltos, critério de enumerar as soluções implicitamente, em um único.

- Incorpora a função objetivo como sendo mais uma restrição.

- Define ao modelar o problema um limite superior I_j, para cada variável X_j, pois a eficiência do algoritmo é inversamente proporcional à ordem do mesmo.

- Resolve problemas de otimização nos quais as variáveis de decisão podem assumir valores inteiros limitados, sem ter que transformá-las em variáveis binárias.

O fluxograma do algoritmo de RÖDDER é apresentado na figura 8.

3.4. Algoritmo de Alvarez. - "LEXSM"

O algoritmo de Alvarez apresenta duas fases para a solução. Na primeira é um algoritmo completo e se transforma em

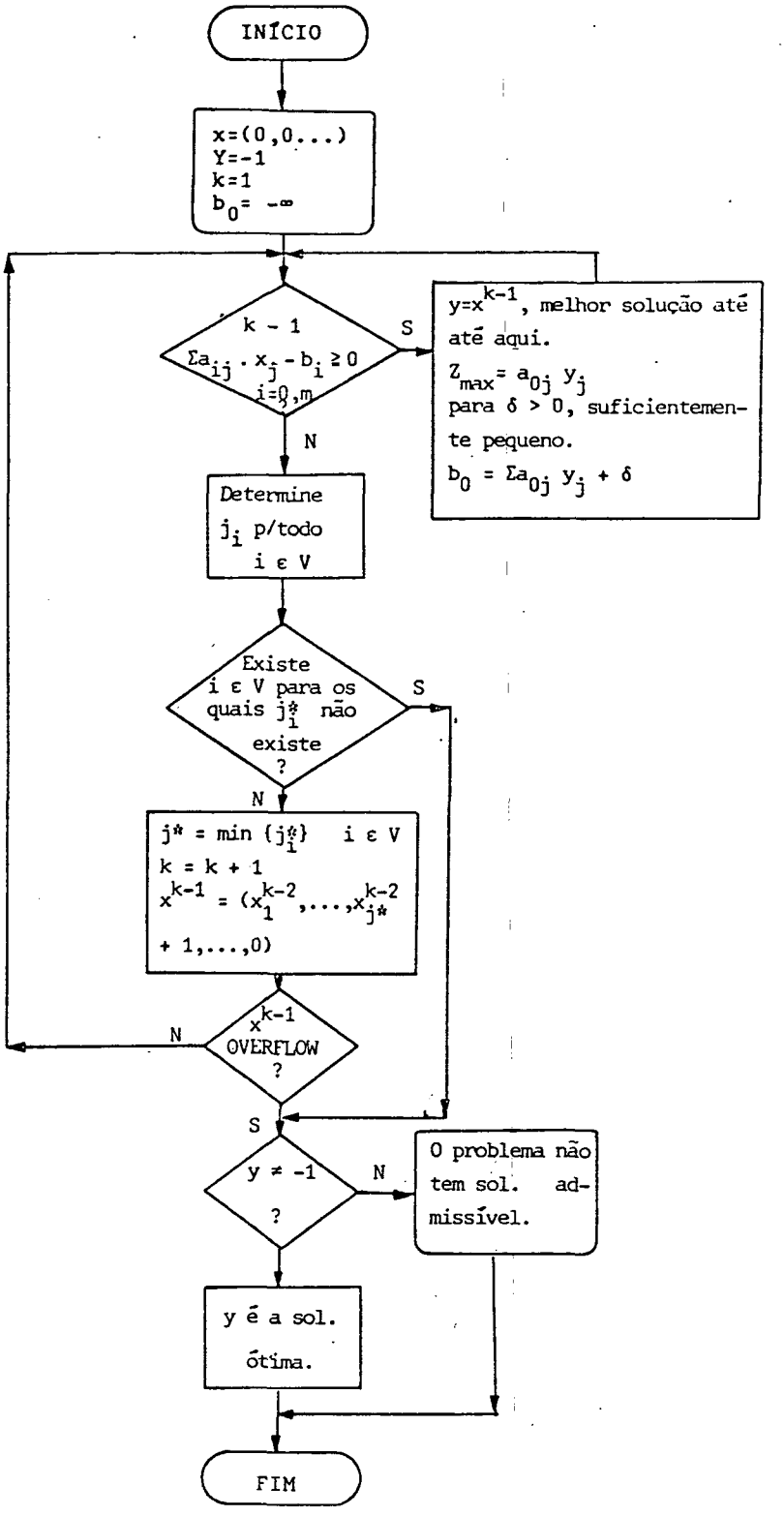


FIGURA 8 - Fluxograma do algoritmo de Rödder.

admissível quando aplicada a segunda fase.

3.4.1. Formulação do Modelo Matemático.

$$\text{Max} \quad Z = \Sigma a_{0j} \cdot X_j \qquad 1 \leq j \leq n$$

$$\text{S.A} \quad \sum_j a_{ij} \cdot X_j \leq b_i \qquad 1 \leq i \leq m \qquad 1 \leq j \leq n$$

Onde:

$$X_j \in \{0, 1\} \qquad 1 \leq j \leq n$$

$$a_{0j} \geq 0 \qquad 1 \leq j \leq n$$

$$a_{0k} \leq a_{0j} \qquad 1 \leq k < j \leq n$$

3.4.2. Modificações introduzidas por ALVAREZ no algoritmo LEXS.

As modificações básicas que deram origem ao algoritmo de ALVAREZ, são:

a) Determinação de uma solução heurística inicial.

Na solução do modelo os algoritmos LEXS e LEXSM incorporam a função objetivo às restrições existentes, como sendo mais uma restrição. Para que a nova restrição não afete a solução do problema o valor inicial de b_0 é definido pela relação:

$$b_0 \leq \Sigma \, |a_{0j}| \cdot I_j \qquad 1 \leq j \leq n$$

portanto para qualquer solução X viável ter-se-á:

$$\Sigma \, a_{0j} \, X_j - b_0 \geq 0 \qquad 1 \leq j \leq n$$

A adição desta nova restrição terá a finalidade de, uma vez encontrada a primeira solução viável, tornar todas as demais soluções viáveis que advirem melhores que a anterior, até a obtenção da solução ótima.

A nova restrição só será ativa após a obtenção de uma

solução viável.

Portanto, se na 1ª fase deste algoritmo obtém-se uma solução inicial viável que produza na função objetivo um valor na vizinhança do ótimo ter-se-á enumerado implicitamente um número elevado de soluções que deixarão de ser testadas na 2ª fase.

Assim sendo, ALVAREZ utilizou 2 idéias heurísticas para a determinação da solução inicial, na 1ª fase do seu algoritmo:

- a heurística do incremento relativo, usada quando existe algum b_i não-positivo (figura 9);
- a heurística de Kochenberger, usada quando os b_i forem naõ negativos (figura 10).

b) Determinação do salto (j_0^*).

O critério de enumerar soluções implicitamente segundo a ordenação lexicografica é denominado de salto.

A partir de uma solução X viável, a nova solução $X^1 > X$ a ser testada será definida por:

$$X_j^1 = 1 \qquad \text{para} \qquad j_0^* < j \leq n$$

onde j_0^* será o primeiro índice i tal que:

$$\Sigma a_{0j} - b_0 > 0 \qquad \text{para} \quad i \leq j \leq n \quad \text{e} \quad 1 \leq i \leq n$$

c) Método de enumeração de uma solução de LEXSM.

A enumeração de uma solução será realizada em duas etapas:

1) Considera-se apenas duas restrições, a restrição pro

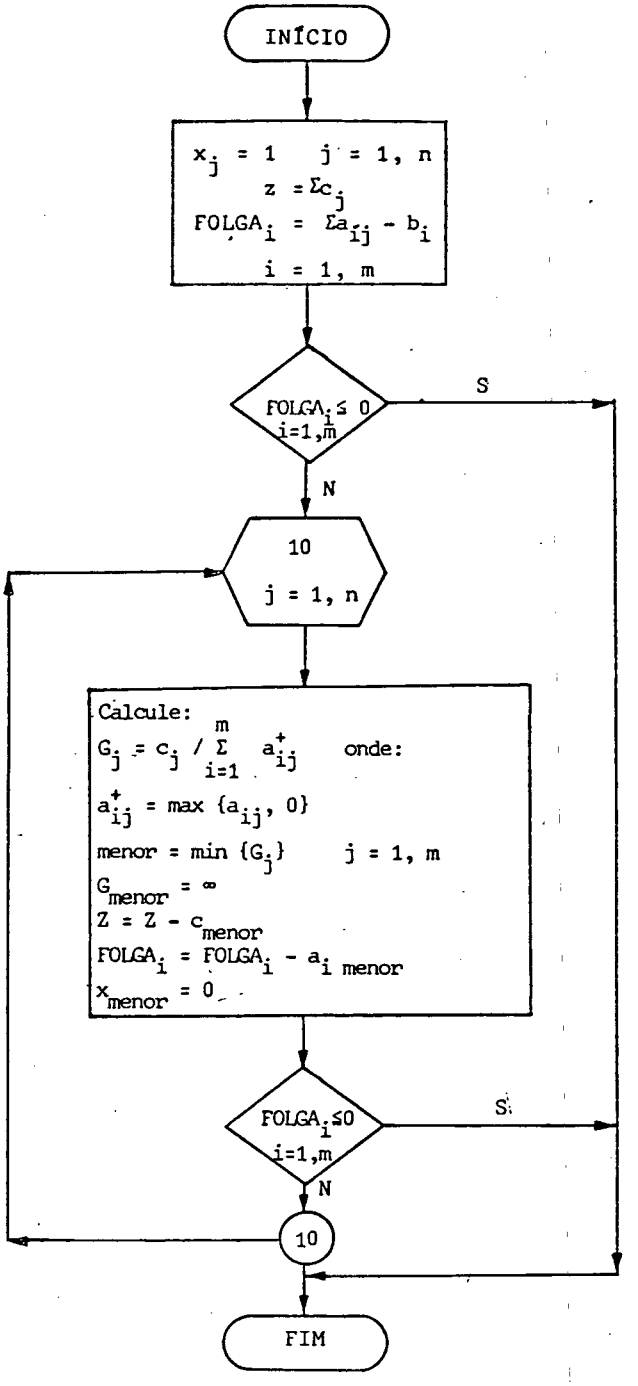


FIGURA 9 - Fluxograma do algoritmo para determinação de uma solução heurística inicial (Heurística do Incremento Relativo).

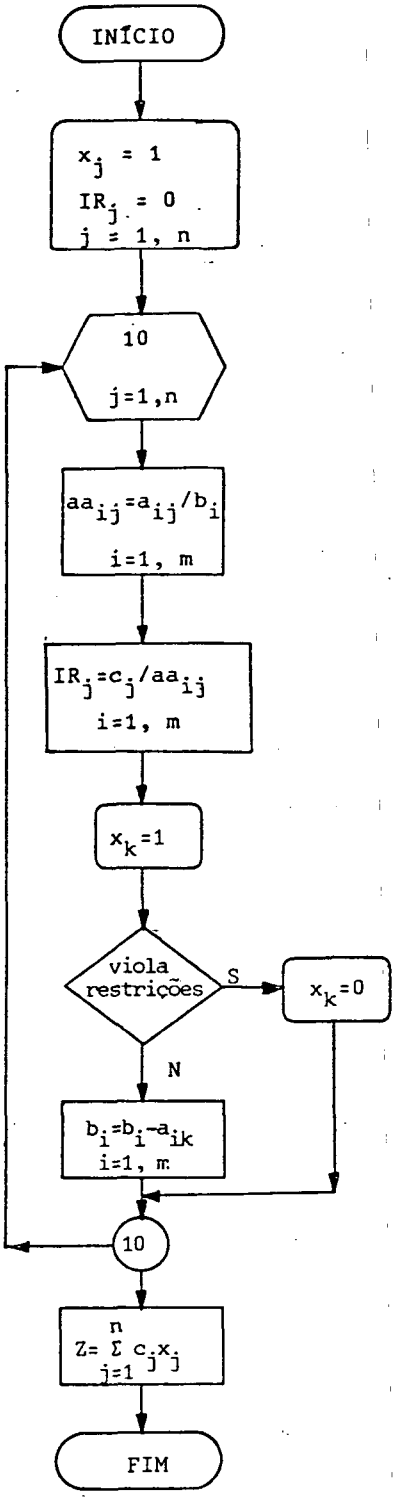


FIGURA 10 - Fluxograma do algoritmo para determinação de uma solução heurística inicial (Heurística de Koshenberger).

veniente da função objetivo e a restrição mais sensível ou de maior folga.

A sensibilidade (folga) entre as restrições será definida para dois casos:

- quando todos os b_i forem positivos, a sensibilidade para a restrição i é dada pela relação:

$$R_i = (\Sigma a_{ij} - b_i)/b_i \quad ; \quad 1 \leq i \leq m \qquad 1 \leq j \leq n$$

Se $\Sigma a_{ij} - b_i \leq 0$ a restrição i será desconsiderada.

- quando houver algum b_i não positivo a sensibilidade será definida por:

$$R_i = b_i - \Sigma a_{ij} \quad ; \quad 1 \leq i \leq m \qquad 1 \leq j \leq n$$

Uma solução somente atingirá a etapa 2 se for viável às duas restrições em consideração da etapa 1.

2) Após a obtenção de uma solução viável na etapa 1, ela será testada nas demais restrições numa ordem decrescente de sensibilidade. Se, ao testar-se a viabilidade da solução em uma restrição, ocorrer a violação da mesma, esta passa a ser a nova restrição flutuante, retornando-se à etapa 1. Se todas as restrições forem verificadas, atualiza-se o valor de b_0, e a restrição flutuante volta a ser a de maior folga, retornando-se à etapa 1.

As modificações descritas acima deram ao algoritmo LEXSM uma performance superior ao do algoritmo LEXS. O fluxograma do algoritmo de ALVAREZ[33] é apresentado nas figuras 11 e 12.

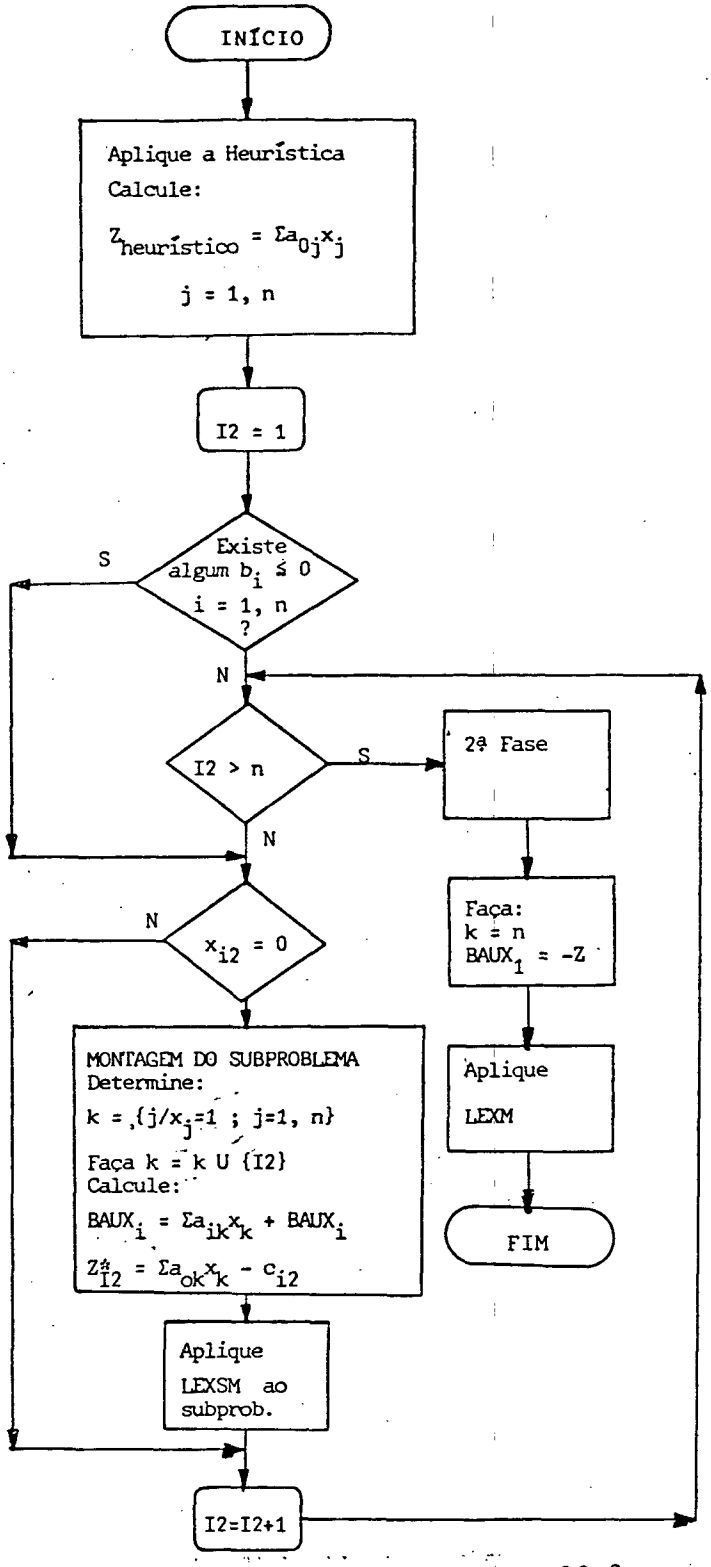


FIGURA 11- Fluxograma do algoritmo de Alvarez - 1ª fase.

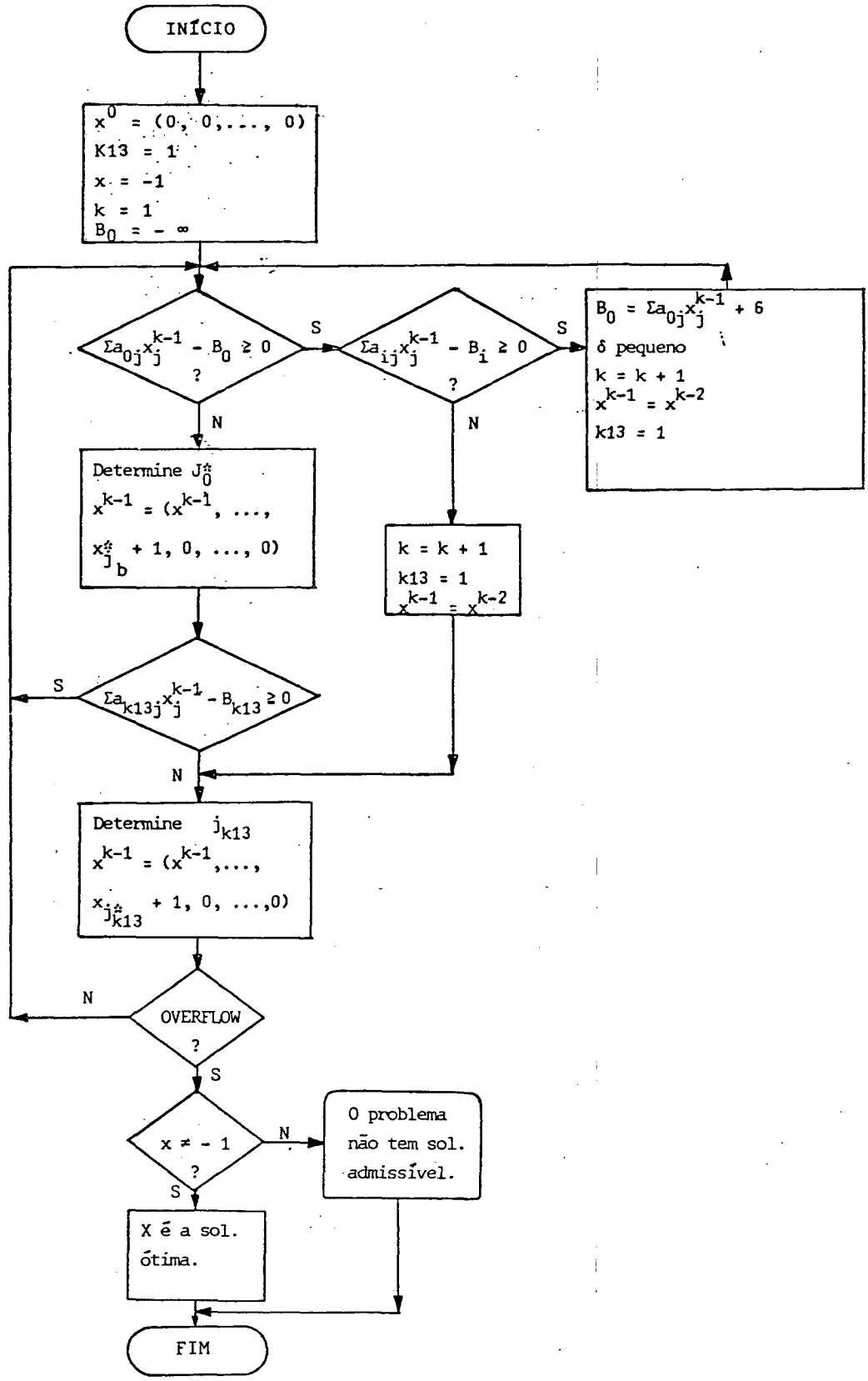


FIGURA 12 - Fluxograma do algoritmo de Alvarez - 2ª fase.

3.5. ADAPTAÇÃO DO ALGORITMO DE ALVAREZ para o problema da mochila "LEXMOCH".

3.5.1. Introdução

O algoritmo adaptado, denominado de "LEXMOCH", é uma variante do algoritmo de ALVAREZ[33] para resolver o problema da mochila. E como tal, segue os critérios de enumeração implícita e lexicográfica respectivamente. Nos testes realizados, o algoritmo se comportou como admissível.

3.5.2. O algoritmo LEXMOCH.

O algoritmo apresentado segue três passos distintos:

a) Transformação do problema para a forma padrão.

O modelo matemático para o problema da mochila foi apresentado no item 1.2. Para transformá-lo para a forma padrão, basta acrescentar a condição:

$$C_k \geq C_j \qquad \text{para} \qquad 1 \leq k < j \leq n$$

Onde k, j e n são inteiros.

b) Cálculo de uma solução heurística inicial.

O tempo de processamento gasto para a solução de um problema, pela aplicação de um algoritmo de enumeração implícita, está numa razão direta da solução inicial situar-se próxima ou não da solução ótima, como ficou evidenciado depois dos diversos testes realizados. Por esta razão, desenvolveu-se um processo heurístico para a determinação da solução inicial.

Este processo heurístico consiste nos passos seguintes:

1) Coloca-se o problema na forma padrão; (conforme item 3.5.2.-a).

2) calcula-se a média aritmética entre os benefícios, através da fórmula:

$$M = \Sigma a_j/n \qquad 1 \leq j \leq n;$$

3) para k variando de 1 até n alocam-se os recursos menores ou iguais à M, da seguinte maneira:

- A cada recurso (k) alocado faz-se $X_k = 1$ e atualiza-se, quantidade ainda disponível para alocações (S), e o valor da função objetivo. Quando o recurso (k) não for alocado, em virtude de ser maior que S ou M, a variável assume o valor zero ($X_k = 0$).

Obtém-se assim uma solução inicial viável. A partir desta, o algoritmo só testará as soluções viáveis que sejam lexicograficamente maiores.

O fluxograma desta heurística é apresentado na figura 13.

c) Processo de enumeração das soluções

Este processo desenvolve-se a partir dos seguintes passos:

1) Obtem-se uma solução inicial viável;

2) percorre-se o vetor X, em ordem decrescente dos índices, até a primeira posição onde a variável assume o valor zero. Esta posição define o valor de j. Torna-se esta variável igual a um ($X_j=1$) e todas as demais, posteriores a esta, agora numa ordem crescente de índices, iguais a zero. Atualizam-se o valor de S e o valor da função objetivo.

Caso não exista nenhuma variável igual a zero, o valor de j será zero e o processo será interrompido;

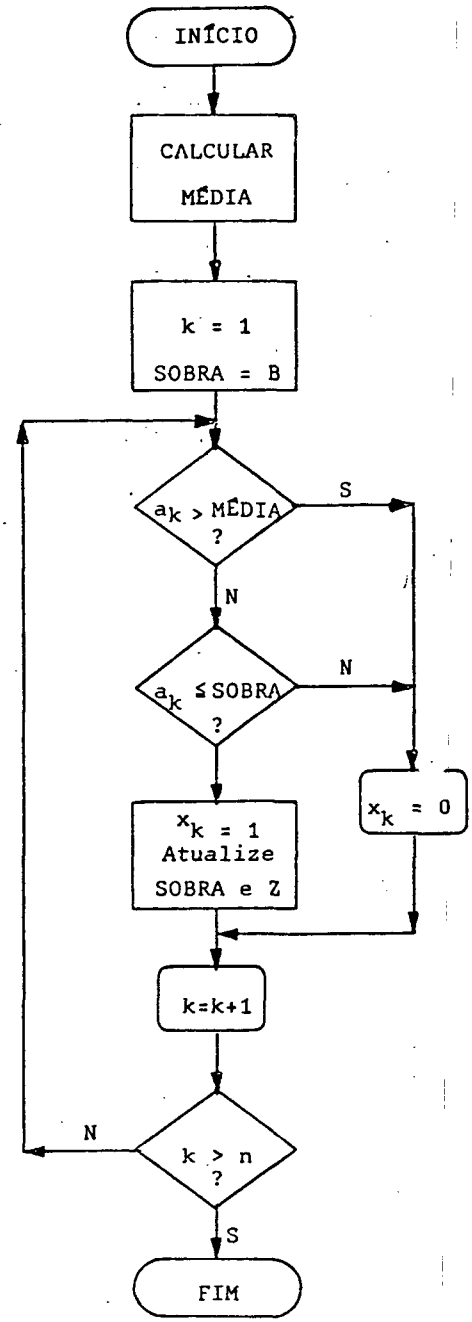


FIGURA 13 - Fluxograma do Algoritmo para determinação de uma solução heurística inicial.

3) testa-se o valor de S.

- se for menor que zero, a nova solução é inviável.

  Recalcula-se j pela relação:

  $j = n - m + 2$ (1)

  Onde: n - número de variáveis

  m - é a posição da última variável de decisão transformada em 1 na solução lexicograficamente maior.

  Retorna-se ao passo 2;

- se for igual a zero, recalcula-se j pela relação (1), testa-se se a nova solução é melhor que a obtida anteriormente, guardando-a se for o caso. Retorna-se ao passo 2;

- se for maior que zero, verifica-se se o total de recursos ainda a alocar, a partir da posição j + 1, é menor ou igual à S. Se for, esta nova solução deve ser comparada com a melhor obtida até o momento. Retorna-se ao passo 2. Se não for menor ou igual à S, verifica-se numa ordem crescente de índices, variando da posição j + 1 até n, a possibilidade de alocação desses recursos, obtendo-se com isto uma nova solução lexicograficamente maior que as anteriormente testadas. Compara-se com a melhor obtida até o momento e retorna-se ao passo 2.

O fluxograma do algoritmo LEXMOCH é apresentado na figura 14.

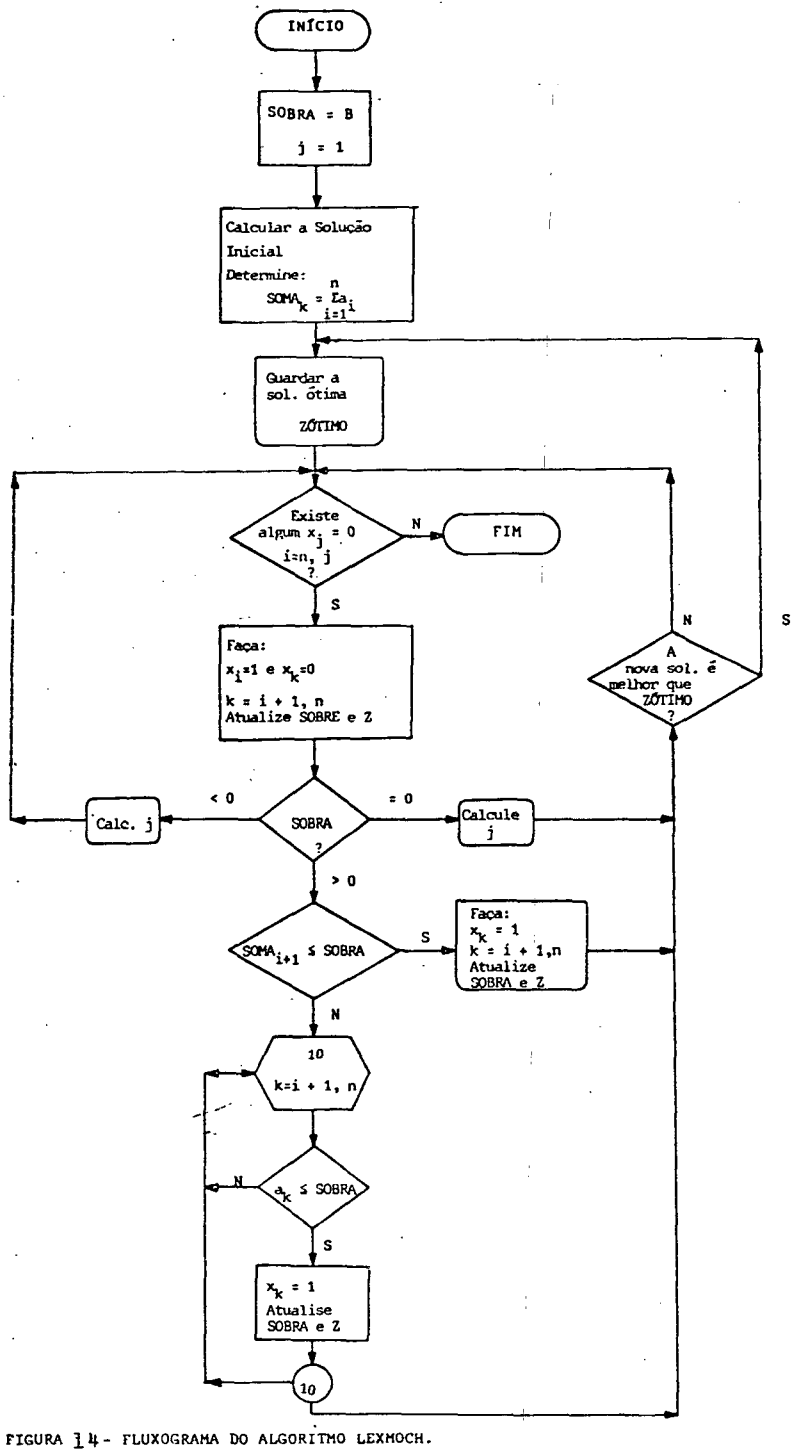


FIGURA 14 - FLUXOGRAMA DO ALGORITMO LEXMOCH.

CAPÍTULO IV

4. ANÁLISE COMPARATIVA

4.1. Introdução

Apresenta-se neste capítulo uma análise comparativa, quanto ao desempenho computacional, entre os algoritmos ZOLTNERS[32] modificado e a variante do algoritmo de ALVAREZ[33] para a solução do problema da mochila "LEXMOCH".

Para tal, os algoritmos foram implementados em linguagem FORTRAN IV, num computador IBM 4341.

Os testes foram realizados em duas etapas a saber:

1) Problemas gerados aleatoriamente, conforme os apresentados por ZOLTNERS[32] em seu artigo.

2) Problemas propostos no artigo de CHVATAL[38].

4.2. Problemas propostos no artigo de ZOLTNERS[32]

Os coeficientes, c_j e a_j, são gerados aleatoriamente seguindo duas distribuições uniformes:

a) $1 < c_j, \ a_j < 100$

b) $1 < c_j, \ a_j < 1000$

O recurso total B é obtido por:

$$B = \alpha \ . \ \Sigma a_j \qquad 1 \leq j \leq n \quad e \quad 0 < \alpha < 1$$

Através de uma rotina, em FORTRAN, montou-se um conjunto de 750 problemas, onde n, número de variáveis, assumiu os valores 5, 10, 15, 20 e 25. Foram realizados os testes para

este conjunto de problemas onde se observou a média, mediana e o tempo máximo de CPU, em milésimos de segundo, conforme apresentado no quadro abaixo.

TABELA 1 - Resultados obtidos na solução dos problemas com os algorítmos de ZOLTNERS modificado e LEXMOCH.

NÚMERO DE VARIÁVEIS	ZOLTNERS MODIFICADO			LEXMOCH		
	MÉDIA	MEDIANA	TEMPO MÁXIMO	MÉDIA	MEDIANA	TEMPO MÁXIMO
5	3	3	7	2	3	7
10	7	2	30	9	7	30
15	12	3	36	96	65	360
20	25	5	70	1382	632	7693
25	29	7	57	37644	14432	355287

Tempo de CPU em milésimos de segundo - IBM 4341 (750 problemas testados).

Do quadro de resultados, da tabela 1, é possível verificar-se a superioridade quanto a eficiência computacional do algorítmo de ZOLTNERS modificado em relação ao LEXMOCH, cujo tempo de processamento cresce de forma exponencial com o número de variáveis.

4.3. Problemas propostos no artigo de CHVÂTAL[38].

Apresentam-se a seguir dois modelos de problemas extraídos do artigo "Hard Knapsack Problems", de CHVÂTAL, e respectivos resultados.

PROBLEMA 1 $\max Z = \Sigma a_j X_j$

tal que $\Sigma a_j X_j \leq (\Sigma a_j/2) \quad 1 \leq j \leq n$

Os valores para a_j seguem uma distribuição uniforme, $1 < a_j < 1000$, inteiros.

Os resultados obtidos na solução de 15 problemas com os dois algoritmos é apresentado no quadro abaixo:

TABELA 2 - Resultados obtidos na solução do problema 1.

NÚMERO DE VARIÁVEIS	ZOLTNERS MODIFICADO			LEXMOCH		
	MÉDIA	MEDIANA	TEMPO MÁXIMO	MÉDIA	MEDIANA	TEMPO MÁXIMO
5	6	7	7	1	2	3
10	153	163	170	20	22	23
15	4440	4270	5010	350	393	490
20	153966	199834	262056	6893	8864	9806
25	*	*	*	130037	149450	189735

Tempo de CPU em milésimo de segundo - IBM 4341.

* Não apresentou solução com 15 minutos de CPU.

Observa-se na tabela 2, que os resultados apresentados pelo algoritmo LEXMOCH são melhores que o de ZOLTNERS modificado. Para os problemas de 25 variáveis o algoritmo de ZOLTNERS modificado não apresentou resultado após 15 minutos de CPU enquanto o LEXMOCH os resolveu em aproximadamente 3 minutos.

PROBLEMA 2 $\max Z = \Sigma a_j X_j$

Tal que $\Sigma a_j X_j \leq B$

Onde $a_j = n \cdot (n + 1) + j$

$$B = \left(\frac{n - 1}{2}\right) \cdot n \cdot (n + 1) + \binom{n}{2}$$

$1 \leq j \leq n$

Para um conjunto de 5 problemas, mostram-se a seguir os resultados obtidos quanto ao tempo de CPU.

TABELA 3 - Resultados obtidos na solução do problema 2.

NÚMERO DE VARIÁVEIS	ZOLTNERS MODIFICADO	LEXMOCH
	TEMPO DE CPU	TEMPO DE CPU
5	10	3
10	167	40
15	5796	1070
20	195027	31318
25	*	904888

Tempo de CPU em milésimo de segundo - IBM 4341.

* não apresentou solução com 30 minutos de CPU.

Observa-se que para os problemas de 25 variáveis o algoritmo de ZOLTNERS modificado não apresentou solução após 30 minutos de CPU. O desempenho computacional de LEXMOCH é superior ao apresentado por ZOLTNERS modificado, apesar de também ter apresentado um crescimento exponencial em tempo de CPU.

Pelos resultados apresentados observa-se que o algoritmo de ZOLTNERS modificado tem um desempenho excelente na solução dos problemas do ítem 4.2. Para os problemas do artigo de CHVÁTAL, o algoritmo LEXMOCH apresentou melhor desempenho.

CAPÍTULO V

5. CONCLUSÕES E SUGESTÕES

O desenvolvimento deste trabalho, onde se faz uma análise comparativa de alguns algoritmos na solução do problema da mochila se caracterizou por três fases distintas.

Na primeira, realizou-se um estudo do artigo de ZOLTNERS[32] que apresenta um algoritmo de busca vertical direta para o problema da mochila. Os resultados apresentados quanto a eficiência computacional eram excelentes e se dizia facilmente implementável.

Após um estudo mais detalhado, observou-se que a apresentação das fases fundamentais do algoritmo era bastante dúbia, o que provocou sérios problemas de interpretação. Tanto assim que a versão final apresentada denominou-se ALGORITMO DE ZOLTNERS MODIFICADO. Os resultados obtidos para os problemas apresentados no artigo, quanto a eficiência computacional, foram muito bons.

Na segunda fase, fez-se um estudo do algoritmo proposto por ALVAREZ[33]; que utiliza a técnica lexicográfica na solução de problemas de programação inteira 0-1.

Apresentou-se um breve histórico dos algoritmos que utilizam a técnica lexicográfica e uma versão do algoritmo de ALVAREZ[33] adaptada a solução do problema da mochila, denominada de LEXMOCH. Os resultados apresentados por esta versão foram muito bons para problemas com até 10 variáveis.

Finalmente, um fato que extrapolou as espectativas

foi observado na fase de testes, pois mostrou-se superior ao LEXMOCH para os problemas do item 4.2, havendo uma inversão para os problemas cujos resultados são apresentados nas tabelas 2 e 3. Este fato ocorreu em virtude do primeiro algoritmo usar como heurística, na determinação da solução inicial, uma ordem decrescente da razão c_j/a_j. Como nos problemas propostos no item 4.3., o valor de c_j é igual a a_j, para todo j, a solução inicial ficou prejudicada. Como o algoritmo LEXMOCH usa na determinação da solução inicial uma ordenação de índices segundo os benefícios numa ordem decrescente de valores, apresentou resultados melhores. É bom ressaltar que o tempo de CPU cresce de forma exponencial devido ao grande número de soluções que devem ser testadas, a partir da solução inicial, com o aumento do número de variáveis. Estudos foram realizados no sentido de melhorar a solução inicial, tendo com isto reduzido o número de testes posteriores, conforme pode-se comprovar com os resultados obtidos.

Para tal, duas propostas de soluções heurísticas iniciais foram testadas.

1) Heurística usada no algoritmo de ZOLTNERS[32].

2) Heurística de KOCHENBERGER para coeficientes positivos, conforme a idéia apresentada por ALVAREZ[33].

Foram implementados em linguagem FORTRAN IV, as duas versões do LEXMOCH com as respectivas heurísticas que tiveram as seguintes denominações:

HIPÓTESE 1 - (H1) - LEXMOCH com a heurística 1.

HIPÓTESE 2 - (H2) - LEXMOCH com a heurística 2.

O algoritmo da hipótese 1 tornou-se competitivo em

tempo de CPU para problemas até 25 variáveis, só que "completo" conforme pode-se observar no quadro da tabela 4. Um dos problemas encontrados foi o da solução inicial ser lexicograficamente maior que a solução ótima, apesar de ter um valor bem próximo dela.

A segunda hipótese, HIPÓTESE 2, quanto ao tempo de processamento é comparável ao do LEXMOCH com um agravante: o algoritmo se transforma em completo, conforme pode-se verificar com os resultados da tabela 4.

Para os testes, foram usados 150 problemas conforme modelo apresentado no item 4.2. e os resultados obtidos são mostrados na tabela 4.

Conclui-se que o algoritmo proposto por ZOLTNERS[32] tem um desempenho computacional excelente, apesar de não se ter a possibilidade de comparar os resultados obtidos na versão apresentada, com os apresentados no artigo, por ter o mesmo usado um computador CDC Cyber 70/74 - 18.

O algoritmo LEXMOCH é competitivo em relação ao ZOLTNERS modificado até dez variáveis podendo ser utilizado como excelente material didático em virtude da sua simplicidade.

Tendo em vista os aspectos levantados neste trabalho, sugere-se como tema para novas pesquisas:

1) Uma heurística para a determinação da solução inicial do algoritmo LEXMOCH;
2) pesquisar sobre os saltos no algoritmo LEXMOCH mantendo o algoritmo admissível;
3) pesquisar uma heurística adicional que permita me-

lhorar o desempenho do algoritmo de ZOLTNERS na solução dos problemas propostos por CHVÁTAL.

TABELA 4 - Resultados obtidos com os algorítmos LEXMOCH, H1 LEXMOCH com a heurística 1 e H2 LEXMOCH com a heurística 2.

Nº DE PRO-	VARIÁVEIS														
	5			10			15			20			25		
BLE- MAS	LEX MOCH	H1	H2	LEX MOCH	H1	H2	Z	H1	H2	LEX MOCH	H1	H2	LEX MOCH	H1	H2
1	261	-	-	529	-	-	4120	-	-	9494	-	-	13216	-	-
2	84	-	-	359	-	-	7801	-	-	8862	-	-	11917	-	-
3	203	-	-	348	-	-	7775	-	-	11161	-	-	10418	-	-
4	214	-	-	496	-	-	8143	-	-	6466	-	-	10399	-	-
5	261	-	-	527	-	-	6579	-	-	8822	-	-	10144	-	-
6	286	-	-	570	-	-	4420	-	-	9287	-	-	8935	-	-
7	209	-	-	318	-	-	6601	-	-	9370	9328	9328	14305	-	-
8	144	-	-	479	-	-	6795	-	-	8734	-	-	8331	-	-
9	297	-	-	451	-	-	6713	-	-	9635	-	-	12923	-	-
10	239	-	-	303	-	-	5464	-	-	9912	-	-	9143	-	-
11	126	-	-	585	-	-	4705	-	-	10036	-	-	12871	-	-
12	192	-	-	313	-	-	7086	-	-	9128	-	-	14610	-	-
13	186	-	-	385	-	-	7666	-	-	9837	-	-	8484	8482	8482
14	217	-	-	323	-	-	6716	-	-	10601	-	-	10284	-	-
15	136	-	-	588	-	-	8049	-	-	11784	-	-	9622	-	-
16	306	-	-	413	-	-	4034	-	-	7437	-	-	7476	-	-
17	120	-	-	457	-	-	5411	-	-	7535	-	-	10551	-	-
18	267	-	-	352	-	-	7786	-	-	7026	-	-	8545	-	-
19	134	-	-	378	-	-	6398	-	-	11155	-	-	8919	-	-
20	189	-	-	431	-	-	5473	-	-	6556	-	-	12072	-	-
21	115	-	-	475	-	-	5401	-	-	19750	-	-	11599	-	-
22	159	-	-	460	-	-	6211	-	-	4805	4709	4709	9284	-	-
23	269	-	-	360	-	-	4088	-	-	8108	-	-	8343	-	-
24	136	-	-	421	-	-	4121	-	-	8904	-	-	12414	-	-
25	119	-	-	478	-	-	8846	-	-	10903	-	-	11000	-	-
26	175	-	-	565	-	-	3822	-	-	10487	-	-	10475	-	-
27	175	-	-	442	-	-	5443	-	-	7468	-	-	13818	-	-
28	130	-	-	601	-	-	8065	8037	8037	9500	-	-	9119	-	-
29	141	-	-	377	-	-	4336	-	-	8201	8098	8098	8909	-	-
30	187	-	-	344	-	-	5794	-	-	7846	-	-	13019	-	-

- a solução encontrada é igual a obtida por LEXMOCH.

BIBLIOGRAFIA

1. HU, T.C., Integer Programming and Network Flows Addison Wesley, Inc., (1969).

2. SALKIN, H.M. and De KLUYVER, C.A., The Knapsack Problem: A SURVEY*, Naval Research Logistics Quarterly 22, 127 - 144 (1975).

3. LORIE, J., and SAVAGE, L.J., "Three Problems in Capital Rationing", Journal of Business (Oct. 1955).

4. WEINGARTNER, H.M., and NESS, D.N., "Methods for the Solution of 0 - 1 Knapsack Problems", presented at the 29th Meeting of Orsa, Santa Monica, California (1966).

5. WEINGARTNER, H.M., "Capital Budgeting and Interre Lated Projects: SURVEY and SYNTHESIS", Management Science 12, 485 - 516 (1968).

6. WEINGARTNER, H.M., Mathematical Programming and the Analysis of Capital Budgeting Problems (Prentice Hall, Inc., 1963).

7. UNGER, V.E., JR., "Capital Budgeting and Mixed Zero-One Integer Programming", AIIE Transactions 11 28-36 (1970).

8. RADHAKRISHNAN, S.R., "Capital Budgeting and Mixed Zero-One Integer Quadratic Programming", Division of Systems and Computer Services, Technical Report, Medical College of Georgia at Augusta (Nov. 1972).

9. MAO, T,C. and WALLINGFOR, B.A., "An Extension of Lawler and

Bell's Method of Discrete Optimization with Example from Capital Budgeting", Management Science 15, 51-60 (1968).

10. BALAS, E., "Duality in Discrete Programming", Department of Operations Research Technical Report Nº 67-5, Stanford University (July 1967).

11. BALAS, E., "Duality in Discrete Programming II; the Quadratic Case", Management Science 16, 14-32 (1969).

12. BALAS, E., "Duality in Discrete Programming III; Nonlinear Objective Function and Constraints", Management Science Research Report Nº 145, Carnegie - Mellun University (Feb.1968).

13. BAUMOL, W.J., and QUANDT, R.E., "Investment and Discount Rates Under Capital Rationing - A Programming Approach", The Economic Journal 75, 317-329 (1965).

14. BYRNE, R., CHARNES, A., COOPER, W.W. and KORTANEK, K.K., "A Chance Constrained Approach to Capital Budgeting with Portfolio Type Payback and Liquidity Constraints and Horizon Posture Controls", Journal of Financial and Quantitative Analysis 11, 339-364 (1967).

15. NASLUND, B., "A Model of Capital Budgeting Under Risk", The Journal of Business (Apr. 1966).

16. WOOLSEY, R.E., "Quick and Dirty Methods in Time Shared Capital Budgeting", Department of Combinatorics and Optimization Research Report Corr 72-8, University of Waterloo (Canada) (Aug. 1971).

17. GLOVER, F., and KLINGMAN, D., "Mathematical Programming Models and Methods for the Journal Selection Problem", Management Science Report Series Report Nº 71-10, Business Research Division, Graduate School of Business Administration, University of Colorado (Dec. 1971).

18. KRAFT, D.H., and HILL, T.W., "The Journal Selection Problem in a University Library System", Research Memorandum Series, Technical Report, School of Industrial Engineering, Purdue University.

19. KRAFT, D.H., and HILL, T.W., "A Lagrangian Formulation of the Journal Selection Model", Technical Report, School of Industrial Engineering, Purdue University (1971).

20. GILMORE, P.C. and GOMORY, R.E., "A Linear Programming Approach to the Cutting Stock Problem I", Operations Research 9, 849-858 (1961).

21. GILMORE, P.C., and GOMORY, R.E., "A Linear Programming Approach to the Cutting Stock Problem II", Operations Research 11, 863-888 (1963).

22. GILMORE, P.C., and GOMORY, R.E., "Multi-stage Cutting Stock Problems of Two and More Dimensions", Operations Research 13, 94-120 (1965).

23. GOMORY, R.E., and JOHNSON, E.L., "Some Continuous Function Related to Corner Polyhedra", IBM Research Report RC3311 (Feb. 23, 1971).

24. MENDES, P.E.P., A alocação Multi-Dimensional: Uma generalização do Problema da Mochila, Compêndio de Teses da PUC

- R.J. - Dep. Eng. Ind., 1975.

25. KOLESAR, P.J., "A Branch and Bound Algorithm for the Knapsack Problem", Management Science, vol. 13, nº 9, May 1967.

26. GREENBERG, H and HEGERICH, R.L., A Branch Search Algorithm for the Knapsack Problem, Management Science, 16, 327-332 (1970).

27. GUIGNARD, M.M., and SPIELBERG, K., Mixed Integer Algorithm for the Zero-One Knapsack Problem, IBM Journal of Research and Development, 16, 424-430 (1972).

28. SHAPIRO, G.F., Dynamic Programming Algorithms for the Integer Programming Problem I: The Integer Programming Problem Viewed as a Knapsack Type Problem, Operations Research vol. 16 (1968).

29. DREYFUS, S.E., An Appraisal of Some Shortest - Path Algorithms, Operations Research, vol. 17, (1969).

30. SHAPIRO, G.F., and WAGNER, H.M., A Finite Renewal algorithm for the Knapsack and Turnpike Models, Operations Research, vol. 14, (1966).

31. GARFINKEL, R.S. NEMHAUSER, G.L., A Survey of Integer Programming Emphasizing Computation and Relations Among Models, Technical Report Nº 156, Dept. of Operations Research, Cornell University, presented at the advanced Seminar on Math. Programming, September 1972, Madison, Wisconsin.

32. ZOLTNERS, A.A., A Direct Descent Binary Knapsack Algorithm, Journal of the Association for Computing Machinery, 25 (2), 304-11, April (1978).

33. ALVAREZ, F. A., DISSERTAÇÃO DE MESTRADO, Um Algoritmo de Programação Linear Inteira Zero-Um Utilizando a Técnica Lexicográfica - (1979) - UFSC.

34. LAWLER, E., and BELL, M., A method for solving discrete optimization problems, Operations Research, 14(6): 1098, 1112, (1966).

35. WALLINGFORD, B.A., and, MAO, J.C.T., An extesion of Lawler and Bell's method of discrete capital budgeting, Management Science, 15(2): Oct. (1968).

36. DRAGAN, I, An algorithme lexicographique pour la resolution des programmes lineaires en variables binaires, Management Science, 16(3): 246-252, Nov. (1969).

37. RÖDDER; Wilhelm. Ein Lexikographischer suchalgorithmus zur ganzzahligen Programmierung: LEXS. Zor, Bd, 20 p. 209-217 (1972).

38. CHVÁTAL, Vasek. Hard Knapsack Problems, McGill University, Montreal, Canada Operations Research, p. 1402-1411 (1980).

ANEXO 1

PROGRAMA DO ALGORITMO DE ZOLTNERS MODIFICADO

FILEO ZOLTMOD FORTRAN A1 VM/SP CONVERSATIONAL MONITOR SYSTEM

```
C*************************************************************************ZOL00010
C***                PROGRAMA  DA   MOCHILA                              ****ZOL00020
C*************************************************************************ZOL00030
      INTEGER A,B,C,X,P,SOBRA,XOTIMO,ZOTIMO,ZBARRA,PINIC,POS,XINIC        ZOL00040
      COMMON /AREA 1/ A(30),NOME(80)                                      ZOL00050
      COMMON /AREA 2/ R(30)                                               ZOL00060
      COMMON /AREA 3/ C(30)                                               ZOL00070
      COMMON /AREA 4/ X(30)                                               ZOL00080
      COMMON /AREA 5/ ZBARRA                                              ZOL00090
      COMMON /AREA 6/ B,SOBRA                                             ZOL00100
      COMMON /AREA 7/ XOTIMO(30),NKUM(30),ZVAR(30)                        ZOL00110
      COMMON /AREA 8/ N                                                   ZOL00120
      COMMON /AREA 9/ LL                                                  ZOL00130
      COMMON /AREA 10/ K                                                  ZOL00140
      COMMON /AREA 11/ ATEMP(30),NUP                                      ZOL00150
C*************************************************************************ZOL00160
C***                LEITURA DOS DADOS                                   ****ZOL00170
C*************************************************************************ZOL00180
      STEMP = 0.                                                          ZOL00190
      READ(5,1)N,NUP                                                      ZOL00200
    1 FORMAT(2I5)                                                         ZOL00210
      DO 5000 ICONT = 1,NUP                                               ZOL00220
      READ(5,2)NOME                                                       ZOL00230
    2 FORMAT(80A1)                                                        ZOL00240
      READ(5,4)(C(I),I=1,N)                                               ZOL00250
    4 FORMAT(16(I5))                                                      ZOL00260
      READ(5,5)(A(I),I=1,N)                                               ZOL00270
    5 FORMAT(16(I5))                                                      ZOL00280
      READ(5,6)B                                                          ZOL00290
    6 FORMAT(I7)                                                          ZOL00300
C*************************************************************************ZOL00310
C***    CALCULO DA RAZAO  -  CUSTO/BENEFICIO     (C/A)                  ****ZOL00320
C*************************************************************************ZOL00330
      CALL STIMER                                                         ZOL00340
      DO 10 L=1,N                                                         ZOL00350
         R(L) = (1.*C(L))/A(L)                                            ZOL00360
   10 CONTINUE                                                            ZOL00370
C*************************************************************************ZOL00380
C***    COLOCAR OS COEFICIENTES EM ORDEM DECRESCENTE SEGUNDO A RAZAO ***ZOL00390
C***                          C/A                                       ***ZOL00400
C*************************************************************************ZOL00410
C                                                                         ZOL00420
      CALL ORDENA                                                         ZOL00430
C                                                                         ZOL00440
C*************************************************************************ZOL00450
C***           INICIALIZACAO    DE     VARIAVEIS                        ***ZOL00460
C*************************************************************************ZOL00470
      ITER = 1                                                            ZOL00480
      SOBRA = 0                                                           ZOL00490
      ZOTIMO = 0                                                          ZOL00500
      P = 0                                                               ZOL00510
C*************************************************************************ZOL00520
C***      VERIFICAR SE A SOLUCAO TRIVIAL EH A OTIMA                     ********ZOL00530
C*************************************************************************ZOL00540
      DO 20 L=1,N                                                         ZOL00550
```

```
FILEO ZOLTMOD  FORTRAN  A1              VM/SP CONVERSATIONAL MONITOR SYSTEM

         X(L) = 0                                                          ZOL00560
         XOTIMO(L) = 0                                                     ZOL00570
         NKUM(L) = 0                                                       ZOL00580
         SOBRA = SOBRA + A(L)                                              ZOL00590
   20 CONTINUE                                                             ZOL00600
      IF(B.EQ.0)GO TO 2000                                                 ZOL00610
C*************************************************************************ZOL00620
C***       VERIFICAR SE A SOLUCAO UNITARIA EH A OTIMA              *******ZOL00630
C*************************************************************************ZOL00640
      IF(SOBRA.GT.B)GO TO 41                                               ZOL00650
      DO 30 L=1,N                                                          ZOL00660
         XOTIMO(L) = 1                                                     ZOL00670
         ZOTIMO = ZOTIMO + C(L)                                            ZOL00680
   30 CONTINUE                                                             ZOL00690
      GO TO 2000                                                           ZOL00700
C*************************************************************************ZOL00710
C***                          INICIO                             ****ZOL00720
C*************************************************************************ZOL00730
   41 SOBRA = B                                                            ZOL00740
      K = 1                                                                ZOL00750
C*************************************************************************ZOL00760
C***              TESTE   DE   VIABILIDADE                       ****ZOL00770
C*************************************************************************ZOL00780
  500 M = K - 1                                                            ZOL00790
      IF(M.EQ.0) GOTO 1000                                                 ZOL00800
      IZ = 0                                                               ZOL00810
      DO 40 L= 1,M                                                         ZOL00820
         IZ = IZ + X(L)*C(L)                                               ZOL00830
   40 CONTINUE                                                             ZOL00840
      ZCHAPE = IZ + C(K)*(1.-SOBRA)/A(K)                                   ZOL00850
      IF(ZOTIMO.LE.ZCHAPE)GOTO 1000                                        ZOL00860
      K = K - 1                                                            ZOL00870
      GO TO 300                                                            ZOL00880
C*************************************************************************ZOL00890
C***       CALCULO DA PRIMEIRA SOLUCAO COMPETITIVA              *******ZOL00900
C*************************************************************************ZOL00910
 1000 IF(A(K).GT.SOBRA)GO TO 32                                            ZOL00920
      X(K) = 1                                                             ZOL00930
      SOBRA = SOBRA - A(K)                                                 ZOL00940
      GO TO 42                                                             ZOL00950
   32 IF(P.EQ.0)P = K                                                      ZOL00960
      IF(SOBRA.EQ.0)GO TO 100                                              ZOL00970
      X(K) = 0                                                             ZOL00980
   42 IF(K.EQ.N)GO TO 100                                                  ZOL00990
      K = K + 1                                                            ZOL01000
      GO TO 500                                                            ZOL01010
  100 CALL ZBAR                                                            ZOL01020
      IF(ZBARRA.GT.ZOTIMO)GO TO 45                                         ZOL01030
      GO TO 300                                                            ZOL01040
C*************************************************************************ZOL01050
C***     SUBSTITUICAO DA MELHOR SOLUCAO COMPETITIVA             ****ZOL01060
C*************************************************************************ZOL01070
   45 ZOTIMO = 0                                                           ZOL01080
      DO 50 L = 1,N                                                        ZOL01090
         XOTIMO(L) = X(L)                                                  ZOL01100
```

```
         ZOTIMO = ZOTIMO + C(L)*XOTIMO(L)                                    ZOL01110
   50 CONTINUE                                                               ZOL01120
C*************************************************************************ZOL01130
C***              R  E  D  U  C  A  O                                    ****ZOL01140
C***     CALCULO DA INFLUENCIA DAS VARIAVEIS NA SOLUCAO                  ****ZOL01150
C***     SE A VARIAVEL FOR ZERO QUAL A INFLUENCIA SE A MESMA FOR UM      ****ZOL01160
C***     E VICE VERSA                                                    ****ZOL01170
C*************************************************************************ZOL01180
   51 LL = P                                                                 ZOL01190
      CALL SOB                                                               ZOL01200
      ZLIN = ZBARRA + C(P) * (1.*SOBRA)/A(P)                                 ZOL01210
      DO 60 L=1,N                                                            ZOL01220
         CBARRA = C(L) - ((1.*C(P))/A(P))*A(L)                               ZOL01230
         IF(CBARRA.LT.0.)CBARRA = -CBARRA                                    ZOL01240
            ZVAR(L) = ZLIN - CBARRA                                          ZOL01250
         IF(ZOTIMO.GE.ZVAR(L))NKUM(L) = 3                                    ZOL01260
   60 CONTINUE                                                               ZOL01270
C*************************************************************************ZOL01280
C***              REGRA  DE PARADA                                       ****ZOL01290
C*************************************************************************ZOL01300
      DO 90 L = P,N                                                          ZOL01310
         IF(NKUM(L).NE.3)GO TO 90                                            ZOL01320
            K = L - 1                                                        ZOL01330
            GO TO 300                                                        ZOL01340
   90 CONTINUE                                                               ZOL01350
      K = N                                                                  ZOL01360
  300 IF(X(K).EQ.0)GO TO 71                                                  ZOL01370
      DO 70 L = 1,K                                                          ZOL01380
         LL = K - L + 1                                                      ZOL01390
         IF(X(LL).EQ.1)GO TO 70                                              ZOL01400
            K = LL                                                           ZOL01410
            GO TO 71                                                         ZOL01420
   70 CONTINUE                                                               ZOL01430
   71 DO 80 L = 1,K                                                          ZOL01440
         LL = K - L + 1                                                      ZOL01450
         IF(X(LL).EQ.0)GO TO 80                                              ZOL01460
         IF(NKUM(LL).NE.3)GO TO 73                                           ZOL01470
   80 CONTINUE                                                               ZOL01480
      GO TO 2000                                                             ZOL01490
C*************************************************************************ZOL01500
C***                    - RETORNO                                       ****ZOL01510
C*************************************************************************ZOL01520
   73 X(LL) = 0                                                              ZOL01530
      P = LL                                                                 ZOL01540
      K = LL + 1                                                             ZOL01550
      ITER = ITER + 1                                                        ZOL01560
      CALL SOB                                                               ZOL01570
      DO 130 L = K,N                                                         ZOL01580
         X(L) = 0                                                            ZOL01590
  130 CONTINUE                                                               ZOL01600
      GO TO 500                                                              ZOL01610
C*************************************************************************ZOL01620
C***      CALCULO DO RECURSO UTILIZADO                                   ****ZOL01630
C*************************************************************************ZOL01640
 2000 IUT = 0                                                                ZOL01650
```

```
FILEO ZOLTMOD  FORTRAN  A1              VM/SP CONVERSATIONAL MONITOR SYSTEM

      DO 120 L =1,N                                                      ZOL01660
         IUT = IUT + A(L)*XOTIMO(L)                                      ZOL01670
  120 CONTINUE                                                           ZOL01680
C*************************************************************************ZOL01690
C***      IMPRESSAO DOS RESULTADOS FINAIS                              ****ZOL01700
C*************************************************************************ZOL01710
      CALL TTIMER(TEMPO)                                                 ZOL01720
      ATEMP(ICONT) = TEMPO                                               ZOL01730
      STEMP = STEMP + TEMPO                                              ZOL01740
      WRITE(6,7)NOME                                                     ZOL01750
      WRITE(7,7)NOME                                                     ZOL01760
    7 FORMAT(1H1,///,80A1)                                               ZOL01770
      WRITE(6,8)(J,J=1,N)                                                ZOL01780
    8 FORMAT(//,T5,'J',T11,15(I5))                                       ZOL01790
      WRITE(6,9)(C(L), L=1,N)                                            ZOL01800
    9 FORMAT(//,T3,'MAX Z= ',T11,16(I5))                                 ZOL01810
      WRITE(6,11)(A(L), L=1,N)                                           ZOL01820
   11 FORMAT(//,T3,'S.A.',T11,16(I5))                                    ZOL01830
      WRITE(6,12)B,IUT                                                   ZOL01840
   12 FORMAT(//,T3,'RECURSO DADO',T23,I7,/,                              ZOL01850
     1          T3,'RECURSO UTILIZADO',T23,I7)                          ZOL01860
      WRITE(6,13)(XOTIMO(L),L=1,N)                                       ZOL01870
   13 FORMAT(//,T3,'X(J)=',T11,16(I5))                                   ZOL01880
      WRITE(6,14)ZOTIMO,ITER,TEMPO                                       ZOL01890
      WRITE(7,14)ZOTIMO,ITER,TEMPO                                       ZOL01900
   14 FORMAT(//,T3,'ZOTIMO=',T11,I8,                                     ZOL01910
     1       //,T3,'NUMERO DE ITERACOES =',T34,I8,                       ZOL01920
     2       //,T3,'TEMPO DE CPU UTILIZADO =',T30,F12.7,2X,'SEGUNDOS')  ZOL01930
 5000 CONTINUE                                                           ZOL01940
      CALL ORDTIM                                                        ZOL01950
      AVG = STEMP/NJP                                                    ZOL01960
      D = (NUP + 1)/2.                                                   ZOL01970
      IZ = IFIX(D)                                                       ZOL01980
       T = D - IZ                                                        ZOL01990
      IF(T.GT.0.0001)GO TO 122                                           ZOL02000
         AMED = ATEMP(IZ)                                                ZOL02010
         GO TO 121                                                       ZOL02020
  122    AMED = (ATEMP(IZ) + ATEMP(IZ+1))/2.                            ZOL02030
  121 AMAX = ATEMP(NUP)                                                  ZOL02040
      WRITE(6,18)N,AVG,AMED,AMAX                                         ZOL02050
   18 FORMAT(1H1,///,T6,65('*'),                                         ZOL02060
     1             /,T7,'N',T23,'MEDIA',T43,'MED',T63,'MAX',           ZOL02070
     2             /,T6,65('*'),                                       ZOL02080
     3             /,T6,I4,T21,F8.5,T41,F8.5,T61,F8.5)                 ZOL02090
      STOP                                                               ZOL02100
      END                                                                ZOL02110
C*************************************************************************ZOL02120
C***      ROTINA PARA ORDENAR DECRESCENTE                              ****ZOL02130
C*************************************************************************ZOL02140
      SUBROUTINE ORDENA                                                  ZOL02150
      INTEGER A,AAUX,ATE,BOLHA,C                                         ZOL02160
      COMMON /AREA 1/ A(30),NOME(80)                                     ZOL02170
      COMMON /AREA 2/ R(30),XINIC(30)                                    ZOL02180
      COMMON /AREA 3/ C(30)                                              ZOL02190
      COMMON /AREA 9/ N                                                  ZOL02200
```

```
      ATE = N -1                                                        ZOL02210
      DO 10 I =1,N                                                      ZOL02220
         ITE = ATE                                                      ZOL02230
         DO 15 L =1,ATE                                                 ZOL02240
            IF(R(L).GE.R(L+1))GO TO 14                                  ZOL02250
               RAUX   = R(L)                                            ZOL02260
               R(L)   = R(L+1)                                          ZOL02270
               R(L+1) = RAUX                                            ZOL02280
               CAUX   = C(L)                                            ZOL02290
               C(L)   = C(L+1)                                          ZOL02300
               C(L+1) = CAUX                                            ZOL02310
               AAUX   = A(L)                                            ZOL02320
               A(L)   = A(L+1)                                          ZOL02330
               A(L+1) = AAUX                                            ZOL02340
               BOLHA = L - 1                                            ZOL02350
               GO TO 15                                                 ZOL02360
   14          ITE = ITE - 1                                            ZOL02370
   15       CONTINUE                                                    ZOL02380
            IF(ITE.EQ.0)RETURN                                          ZOL02390
            IF(BOLHA.EQ.0)RETURN                                        ZOL02400
               ATE = BOLHA                                              ZOL02410
   10 CONTINUE                                                          ZOL02420
      RETURN                                                            ZOL02430
      END                                                               ZOL02440
C*******************************************************************************ZOL02450
C***      ROTINA PARA O CALCULO DE ZBARRA                              ****ZOL02460
C*******************************************************************************ZOL02470
      SUBROUTINE ZBAR                                                   ZOL02480
      INTEGER ZBARRA,X,C                                                ZOL02490
      COMMON /AREA 3/ C(30)                                             ZOL02500
      COMMON /AREA 4/ X(30)                                             ZOL02510
      COMMON /AREA 5/ ZBARRA                                            ZOL02520
      COMMON /AREA 8/ N                                                 ZOL02530
      COMMON /AREA 10/ K                                                ZOL02540
      ZBARRA = 0                                                        ZOL02550
      DO 10 L =1,K                                                      ZOL02560
         ZBARRA = ZBARRA + X(L)*C(L)                                    ZOL02570
   10 CONTINUE                                                          ZOL02580
      RETURN                                                            ZOL02590
      END                                                               ZOL02600
C*******************************************************************************ZOL02610
C***      ROTINA PARA O CALCULO DE SOBRA                               ****ZOL02620
C*******************************************************************************ZOL02630
      SUBROUTINE SOB                                                    ZOL02640
      INTEGER A,X,SOBRA,B,PINIC                                         ZOL02650
      COMMON /AREA 1/ A(30),NOME(80)                                    ZOL02660
      COMMON /AREA 4/ X(30)                                             ZOL02670
      COMMON /AREA 6/ B,SOBRA                                           ZOL02680
      COMMON /AREA 9/ LL                                                ZOL02690
      SOBRA = B                                                         ZOL02700
      DO 10 L =1,LL                                                     ZOL02710
         SOBRA = SOBRA - X(L)*A(L)                                      ZOL02720
   10 CONTINUE                                                          ZOL02730
      RETURN                                                            ZOL02740
      END                                                               ZOL02750
```

FILEO ZOLTMOD FORTRAN A1 VM/SP CONVERSATIONAL MONITOR SYSTEM

```
C*************************************************************************ZOL02760
C***     ROTINA PARA ORDENAR CRESCENTE O TEMPO                          ***ZOL02770
C*************************************************************************ZOL02780
      SUBROUTINE ORDTIM                                                   ZOL02790
      INTEGER ATE,BOLHA                                                   ZOL02800
      COMMON /AREA 11/ ATEMP(30),NUP                                      ZOL02810
      ATE = NUP - 1                                                       ZOL02820
      DO 10 I =1,NUP                                                      ZOL02830
         ITE = ATE                                                        ZOL02840
         DO 15 L =1,ATE                                                   ZOL02850
            IF(ATEMP(L).LE.ATEMP(L+1))GO TO 14                            ZOL02860
               ZAUX   = ATEMP(L)                                          ZOL02870
               ATEMP(L) = ATEMP(L+1)                                      ZOL02880
               ATEMP(L+1) = ZAUX                                          ZOL02890
               BOLHA = L - 1                                              ZOL02900
               GO TO 15                                                   ZOL02910
   14          ITE = ITE - 1                                              ZOL02920
   15       CONTINUE                                                      ZOL02930
            IF(ITE.EQ.0)RETURN                                            ZOL02940
            IF(BOLHA.EQ.0)RETURN                                          ZOL02950
               ATE = BOLHA                                                ZOL02960
   10 CONTINUE                                                           ZOL02970
      RETURN                                                             ZOL02980
      END                                                                ZOL02990
```

ANEXO 2

PROGRAMA DO ALGORITMO DE ALVAREZ ADAPTADO PARA O PROBLEMA DA MOCHILA (LEXMOCH)

```
FILEO LEXMOCH  FORTRAN  A1              VM/SP CONVERSATIONAL MONITOR SYSTEM

C*************************************************************************LEX00010
C***              PROGRAMA  DA   MOCHILA   LEXICO                       ****LEX00020
C*************************************************************************LEX00030
      INTEGER NOME(80),SOMA(50),XOTIMO(50),X(50),A,B,C,SOBRA,ZOTIMO,ZLINLEX00040
      COMMON /AREA1/ A(50),C(50),N                                        LEX00050
      COMMON /AREA2/ ATEMP(50),NUP                                        LEX00060
C*************************************************************************LEX00070
C***              LEITURA DOS DADOS                                     ****LEX00080
C*************************************************************************LEX00090
      READ(5,1)N,NUP                                                      LEX00100
    1 FORMAT(215)                                                         LEX00110
      STEMP = 0.                                                          LEX00120
      DO 5000 ICONT = 1,NUP                                               LEX00130
      READ(5,2)NOME                                                       LEX00140
    2 FORMAT(80A1)                                                        LEX00150
      READ(5,4)(C(I),I=1,N)                                               LEX00160
    4 FORMAT(16(I5))                                                      LEX00170
      READ(5,5)(A(I),I=1,N)                                               LEX00180
    5 FORMAT(16(I5))                                                      LEX00190
      READ(5,6)B                                                          LEX00200
    6 FORMAT(I7)                                                          LEX00210
C*************************************************************************LEX00220
C***                        INICIO                                     ***LEX00230
C*************************************************************************LEX00240
C                                                                         LEX00250
      CALL STIMER                                                         LEX00260
C                                                                         LEX00270
C*************************************************************************LEX00280
C***    COLOCAR OS COEFICIENTES DA FUNCAO OBJETIVO EM ORDEM            ***LEX00290
C***                        CRESCENTE                                  ***LEX00300
C*************************************************************************LEX00310
C                                                                         LEX00320
      CALL ORDENA                                                         LEX00330
C                                                                         LEX00340
C*************************************************************************LEX00350
C***           INICIALIZACAO    DE     VARIAVEIS                       ***LEX00360
C*************************************************************************LEX00370
      ITER = 1                                                            LEX00380
      SOBRA = B                                                           LEX00390
      NSOMA = 0                                                           LEX00400
      ZOTIMO = 0                                                          LEX00410
      MARCA = 0                                                           LEX00420
      J = 1                                                               LEX00430
      DO 10 L =1,N                                                        LEX00440
         X(L) = 0                                                         LEX00450
   10 CONTINUE                                                            LEX00460
C*************************************************************************LEX00470
C***      CALCULO DO VETOR SOMA PARCIAL                                ******LEX00480
C*************************************************************************LEX00490
      DO 20 L =1,N                                                        LEX00500
         K = N-L+1                                                        LEX00510
         NSOMA = NSOMA + A(K)                                             LEX00520
         SOMA(K) = NSOMA                                                  LEX00530
   20 CONTINUE                                                            LEX00540
C*************************************************************************LEX00550
```

```
FILEO LEXMOCH  FORTRAN  A1              VM/SP CONVERSATIONAL MONITOR SYSTEM

C***         SOLUCAO INICIAL                                          ****LEX00560
C*************************************************************************LEX00570
      ZLIN = 0                                                           LEX00580
      AMEDIA = (FLOAT(NSOMA)/N)                                          LEX00590
      MEDIA = AMEDIA                                                     LEX00600
      DO 90 L = 1,N                                                      LEX00610
         K = N - L + 1                                                   LEX00620
         IF(A(K).GT.MEDIA)GO TO 90                                       LEX00630
         IF(A(K).GT.SOBRA)GO TO 90                                       LEX00640
            X(K) = 1                                                     LEX00650
            ZLIN = ZLIN + C(K)                                           LEX00660
            SOBRA = SOBRA - A(K)                                         LEX00670
   90 CONTINUE                                                           LEX00680
C*************************************************************************LEX00690
C***     SUBSTITUICAO DA MELHOR SOLUCAO COMPETITIVA                      ****LEX00700
C*************************************************************************LEX00710
      IF(ZLIN.LT.ZOTIMO)GO TO 31                                         LEX00720
   21 ZOTIMO = ZLIN                                                      LEX00730
      DO 30 L = 1,N                                                      LEX00740
         XOTIMO(L) = X(L)                                                LEX00750
   30 CONTINUE                                                           LEX00760
C     WRITE(6,16)(X(L),L=1,N),ZLIN                                       LEX00770
C  16 FORMAT(/,5X,25I3,3X,'ZLIN',2X,I5)                                  LEX00780
C*************************************************************************LEX00790
C***      CONSTRUCAO DAS SOLUCOES LEXICOGRAFICAS MAIORES QUE        **** LEX00800
C***                    A SOLUCAO INICIAL                           **** LEX00810
C*************************************************************************LEX00820
   31 DO 40 L = J,N                                                      LEX00830
         M = N - L + 1                                                   LEX00840
         IF(X(M).EQ.0)GO TO 41                                           LEX00850
   40 CONTINUE                                                           LEX00860
      GO TO 2000                                                         LEX00870
   41 X(M) = 1                                                           LEX00880
      ZLIN = ZLIN + C(M)                                                 LEX00890
      SOBRA = SOBRA - A(M)                                               LEX00900
      IM = M + 1                                                         LEX00910
      IF(IM.GT.N)GO TO 51                                                LEX00920
      DO 50 L = IM,N                                                     LEX00930
         ZLIN = ZLIN - C(L)*X(L)                                         LEX00940
         SOBRA = SOBRA + A(L)*X(L)                                       LEX00950
         X(L) = 0                                                        LEX00960
   50 CONTINUE                                                           LEX00970
C*************************************************************************LEX00980
C***      CALCULO DO PULO                                                ****LEX00990
C*************************************************************************LEX01000
   51 IF(SOBRA)77,88,99                                                  LEX01010
   77 ITER = ITER + 1                                                    LEX01020
      J = N - M + 2                                                      LEX01030
      X(M) = 0                                                           LEX01040
      ZLIN = ZLIN - C(M)                                                 LEX01050
      SOBRA = SOBRA + A(M)                                               LEX01060
      GO TO 31                                                           LEX01070
   88 J=N-M+2                                                            LEX01080
      GO TO 87                                                           LEX01090
   99 IF(IM.LE.N)GO TO 98                                                LEX01100
```

```
FILEO LEXMOCH  FORTRAN  A1              VM/SP CONVERSATIONAL MONITOR SYSTEM

      J = 2                                                              LEX01110
      GO TO 87                                                           LEX01120
   98 IF(SOMA(IM).LE.SOBRA)GO TO 63                                      LEX01130
      IS = N - IM + 1                                                    LEX01140
      DO 170 L=1,IS                                                      LEX01150
         K = N-L+1                                                       LEX01160
         IF(A(K).GT.SOBRA)GO TO 169                                      LEX01170
          X(K) = 1                                                       LEX01180
          ZLIN = ZLIN + C(K)                                             LEX01190
          SOBRA = SOBRA - A(K)                                           LEX01200
  169    CONTINUE                                                        LEX01210
  170    CONTINUE                                                        LEX01220
      J = 1                                                              LEX01230
      GO TO 87                                                           LEX01240
   63 DO 70 L = IM,N                                                     LEX01250
      X(L) = 1                                                           LEX01260
      ZLIN = ZLIN + C(L)                                                 LEX01270
      SOBRA = SOBRA - A(L)                                               LEX01280
   70 CONTINUE                                                           LEX01290
      J = N-IM+2                                                         LEX01300
   87 ITER = ITER + 1                                                    LEX01310
      IF(ZLIN.GT.ZOTIMO)GO TO 21                                         LEX01320
      GO TO 31                                                           LEX01330
C*************************************************************************LEX01340
C***     CALCULO DO RECURSO UTILIZADO                                  ****LEX01350
C*************************************************************************LEX01360
 2000 IUT = 0                                                            LEX01370
      DO 130 L =1,N                                                      LEX01380
         IUT = IUT + A(L)*XOTIMO(L)                                      LEX01390
  130 CONTINUE                                                           LEX01400
C*************************************************************************LEX01410
C***     IMPRESSAO DOS RESULTADOS FINAIS                               ****LEX01420
C*************************************************************************LEX01430
      CALL TTIMER(TEMPO)                                                 LEX01440
      ATEMP(ICONT) = TEMPO                                               LEX01450
      STEMP = STEMP+ TEMPO                                               LEX01460
      WRITE(6,7)NOME                                                     LEX01470
      WRITE(7,7)NOME                                                     LEX01480
    7 FORMAT(1H1,///,80A1)                                               LEX01490
      WRITE(6,8)(J,J=1,N)                                                LEX01500
    8 FORMAT(//,T5,'J',T11,16(I5))                                       LEX01510
      WRITE(6,9)(C(L), L=1,N)                                            LEX01520
    9 FORMAT(//,T3,'MAX Z= ',T11,16(I5))                                 LEX01530
      WRITE(6,11)(A(L), L=1,N)                                           LEX01540
   11 FORMAT(//,T3,'S.A.',T11,16(I5))                                    LEX01550
      WRITE(6,12)3,IUT                                                   LEX01560
   12 FORMAT(//,T3,'RECURSO DADO',T23,I7,/,                              LEX01570
     1           T3,'RECURSO UTILIZADO',T23,17)                          LEX01580
      WRITE(6,13)(XOTIMO(L),L=1,N)                                       LEX01590
   13 FORMAT(//,T3,'X(J)=',T11,16(I5))                                   LEX01600
      WRITE(6,14)ZOTIMO,ITER,TEMPO                                       LEX01610
      WRITE(7,14)ZOTIMO,ITER,TEMPO                                       LEX01620
   14 FORMAT(//,T3,'ZOTIMO=',T11,I8,                                     LEX01630
     1       //,T3,'NUMERO DE ITERACOES =',T34,I6,                       LEX01640
     2       //,T3,'TEMPO DE CPU UTILIZADO =',T30,F12.7,2X,'SEGUNDOS')  LEX01650
```

```
 5000 CONTINUE                                                          LEX01660
      CALL ORDTIM                                                       LEX01670
      AVG = STEMP/NUP                                                   LEX01680
      D = (NUP + 1)/2.                                                  LEX01690
      IZ = IFIX(D)                                                      LEX01700
      T = D - IZ                                                        LEX01710
      IF(D.GT.0.0001)GO TO 122                                          LEX01720
         AMED = ATEMP(IZ)                                               LEX01730
         GO TO 123                                                      LEX01740
  122    AMED = (ATEMP(IZ) + ATEMP(IZ+1))/2.                           LEX01750
  123 AMAX = ATEMP(NUP)                                                 LEX01760
      WRITE(6,18)N,AVG,AMED,AMAX                                        LEX01770
   18 FORMAT(1H1,///,T6,65('*'),                                        LEX01780
     1             /,T7,'N',T23,'MEDIA',T43,'MED',T63,'MAX',          LEX01790
     2             /,T6,65('*'),                                      LEX01800
     3             /,T6,I4,T21,F9.5,T41,F9.5,T61,F9.5)                LEX01810
      STOP                                                              LEX01820
      END                                                               LEX01830
C*************************************************************************LEX01840
C***      ROTINA PARA ORDENAR CRESCENTE                              ****LEX01850
C*************************************************************************LEX01860
      SUBROUTINE ORDENA                                                 LEX01870
      INTEGER A,AAUX,ATE,BOLHA,C,CAUX,ZAUX                              LEX01880
      COMMON /AREA1/ A(50),C(50),N                                      LEX01890
      ATE = N -1                                                        LEX01900
      DO 10 I =1,N                                                      LEX01910
         ITE = ATE                                                      LEX01920
         DO 15 L =1,ATE                                                 LEX01930
            IF(C(L).GE.C(L+1))GO TO 14                                  LEX01940
               CAUX   = C(L)                                            LEX01950
               C(L)   = C(L+1)                                          LEX01960
               C(L+1) = CAUX                                            LEX01970
               AAUX   = A(L)                                            LEX01980
               A(L)   = A(L+1)                                          LEX01990
               A(L+1) = AAUX                                            LEX02000
               BOLHA = L - 1                                            LEX02010
               GO TO 15                                                 LEX02020
   14          ITE = ITE - 1                                           LEX02030
   15       CONTINUE                                                    LEX02040
            IF(ITE.EQ.0)RETURN                                          LEX02050
            IF(BOLHA.EQ.0)RETURN                                        LEX02060
               ATE = BOLHA                                              LEX02070
   10 CONTINUE                                                          LEX02080
      RETURN                                                            LEX02090
      END                                                               LEX02100
C*************************************************************************LEX02110
C***      ROTINA PARA ORDENAR CRESCENTE O TEMPO                      ****LEX02120
C*************************************************************************LEX02130
      SUBROUTINE ORDTIM                                                 LEX02140
      INTEGER ATE,BOLHA                                                 LEX02150
      COMMON /AREA2/ ATEMP(50),NUP                                      LEX02160
      ATE = NUP - 1                                                     LEX02170
      DO 10 I =1,NUP                                                    LEX02180
         ITE = ATE                                                      LEX02190
         DO 15 L =1,ATE                                                 LEX02200
```

```
            IF(ATEMP(L).LE.ATEMP(L+1))GO TO 14                          LEX02210
               ZAUX   = ATEMP(L)                                        LEX02220
               ATEMP(L) = ATEMP(L+1)                                    LEX02230
               ATEMP(L+1) = ZAUX                                        LEX02240
               BOLHA = L - 1                                            LEX02250
               GO TO 15                                                 LEX02260
   14          ITE = ITE - 1                                            LEX02270
   15       CONTINUE                                                    LEX02280
            IF(ITE.EQ.0)RETURN                                          LEX02290
            IF(BOLHA.EQ.0)RETURN                                        LEX02300
               ATE = BOLHA                                              LEX02310
   10 CONTINUE                                                          LEX02320
      RETURN                                                            LEX02330
      END                                                               LEX02340
```

ANEXO 3

PROBLEMAS PROPOSTOS NO ARTIGO DE CHVATAL

A) PROBLEMA 1 - ALGORITMO ZOLTNERS MODIFICADO

*** PROBLEMA 1 ***

J	1	2	3	4	5
MAX Z=	69	94	15	83	16
S.A.	69	94	15	83	16

RECURSO DADO 138
RECURSO UTILIZADO 125

X(J)= 0 1 1 0 1

ZOTIMO= 125

NUMERO DE ITERACOES = 5

TEMPO DE CPU UTILIZADO = 0.0033331 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5
MAX Z=	92	36	7	93	82
S.A.	92	36	7	93	82

RECURSO DADO 155
RECURSO UTILIZADO 136

X(J)= 0 1 1 1 0

ZOTIMO= 136

NUMERO DE ITERACOES = 12

TEMPO DE CPU UTILIZADO = 0.0066662 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5
MAX Z=	81	15	66	65	18
S.A.	81	15	66	65	18

RECURSO DADO 122
RECURSO UTILIZADO 114

X(J) = 1 1 0 0 1

ZOTIMO= 114

NUMERO DE ITERACOES = 6

TEMPO DE CPU UTILIZADO = 0.0066662 SEGUNDOS

*** PROBLEMA 4 ***

J	1	2	3	4	5
MAX Z=	40	74	78	91	37
S.A.	40	74	78	91	37

RECURSO DADO 160
RECURSO UTILIZADO 155

X(J)= 1 0 1 0 1

ZOTIMO= 155

NUMERO DE ITERACOES = 8

TEMPO DE CPU UTILIZADO = 0.0066662 SEGUNDOS

*** PROBLEMA 5 ***

J	1	2	3	4	5
MAX Z=	67	41	10	76	32
S.A.	67	41	10	76	32

RECURSO DADO 113
RECURSO UTILIZADO 109

X(J)=	1	0	1	0	1

ZOTIMO= 109

NUMERO DE ITERACOES = 8

TEMPO DE CPU UTILIZADO = 0.0066662 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	69	94	15	83	16	92	36	7	93	82
S.A.	69	94	15	83	16	92	36	7	93	82

RECURSO DADO 293
RECURSO UTILIZADO 293

X(J)=	1	1	1	0	1	1	0	1	0	0

ZOTIMO= 293

NUMERO DE ITERACOES = 260

TEMPO DE CPU UTILIZADO = 0.1699890 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	81	15	66	65	18	40	74	78	91	37
S.A.	81	15	66	65	18	40	74	78	91	37

RECURSO DADO 282
RECURSO UTILIZADO 282

X(J)=	1	1	1	1	1	0	0	0	0	1

ZOTIMO= 282

NUMERO DE ITERACOES = 222

TEMPO DE CPU UTILIZADO = 0.1633223 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	67	41	10	76	32	95	29	29	36	44
S.A.	67	41	10	76	32	95	29	29	36	44

RECURSO DADO 229
RECURSO UTILIZADO 229

X(J)=	1	0	1	1	1	0	0	0	0	1

ZOTIMO= 229

NUMERO DE ITERACOES = 175

TEMPO DE CPU UTILIZADO = 0.1266585 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	697	949	155	837	167	921	366	72	932	826	816	153	660	650	181
S.A.	697	949	155	837	167	921	366	72	932	826	816	153	660	650	181

RECURSO DADO 4191
RECURSO UTILIZADO 4191

X(J)=	1	1	0	0	0	0	0	1	0	1	1	0	0	1	1

ZOTIMO= 4191

NUMERO DE ITERACOES = 3913

TEMPO DE CPU UTILIZADO = 4.0397406 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	404	748	781	918	379	673	413	107	765	326	957	290	295	365	440
S.A.	404	748	781	918	379	673	413	107	765	326	957	290	295	365	440

RECURSO DADO 3930
RECURSO UTILIZADO 3930

X(J)=	1	1	1	0	1	0	1	0	1	0	0	0	0	0	1

ZOTIMO= 3930

NUMERO DE ITERACOES = 4532

TEMPO DE CPU UTILIZADO = 4.2697258 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	323	646	552	919	212	353	99	883	961	710	14	657	415	679	440
S.A.	323	646	552	919	212	353	99	883	961	710	14	657	415	679	440

RECURSO DADO 3931
RECURSO UTILIZADO 3931

X(J)=	1	1	1	0	0	0	1	0	1	0	1	1	0	1	0

ZOTIMO= 3931

NUMERO DE ITERACOES = 5469

TEMPO DE CPU UTILIZADO = 5.0096783 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													
MAX Z=	697	949	155	837	167	921	366	72	932	826	816	153	660	650	181	404
748	781	918	379													
S.A.	697	949	155	837	167	921	366	72	932	826	816	153	660	650	181	404
748	781	918	379													

RECURSO DADO 5806
RECURSO UTILIZADO 5806

X(J)=	1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0
0	0	0	0													

ZOTIMO= 5806

NUMERO DE ITERACOES = 161672

TEMPO DE CPU UTILIZADO = 199.833847 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													
MAX Z=	673	413	107	765	326	957	290	295	365	440	323	646	552	919	212	353
99	883	961	710													
S.A.	673	413	107	765	326	957	290	295	365	440	323	646	552	919	212	353
99	883	961	710													

RECURSO DADO 5144
RECURSO UTILIZADO 5144

X(J)=	1	1	1	1	1	1	1	0	0	1	0	0	0	0	1	0
0	0	1	0													

ZOTIMO= 5144

NUMERO DE ITERACOES = 229224

TEMPO DE CPU UTILIZADO = 262.056396 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													
MAX Z=	14	657	415	679	440	814	105	793	4	578	88	67	156	230	794	901
517	514	868	987													
S.A.	14	657	415	679	440	814	105	793	4	578	88	67	156	230	794	901
517	514	868	987													

RECURSO DADO 4810
RECURSO UTILIZADO 4810

X(J)=	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	0													

ZOTIMO= 4810

NUMERO DE ITERACOES = 1

TEMPO DE CPU UTILIZADO = 0.0066662 SEGUNDOS

B) PROBLEMA 1 - ALGORITMO LEXMOCH

*** PROBLEMA 1 ***

J	1	2	3	4	5
MAX Z=	94	83	69	16	15
S.A.	94	83	69	16	15

RECURSO DADO 138
RECURSO UTILIZADO 125

X(J) = 1 0 0 1 1

ZOTIMO= 125

NUMERO DE ITERACOES = 7

TEMPO DE CPU UTILIZADO = 0.0 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5
MAX Z=	93	92	82	36	7
S.A.	93	92	82	36	7

RECURSO DADO 155
RECURSO UTILIZADO 136

X(J) = 1 0 0 1 1

ZOTIMO= 136

NUMERO DE ITERACOES = 7

TEMPO DE CPU UTILIZADO = 0.0 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5
MAx Z=	81	66	65	18	15
S.A.	81	66	65	18	15

RECURSO DADO 122
RECURSO UTILIZADO 114

X(J)= 1 0 0 1 1

ZOTIMO= 114

NUMERO DE ITERACOES = 7

TEMPO DE CPU UTILIZADO = 0.0033331 SEGUNDOS

*** PROBLEMA 4 ***

J	1	2	3	4	5
MAX Z=	91	78	74	40	37
S.A.	91	78	74	40	37

RECURSO DADO 160
RECURSO UTILIZADO 155

X(J) = 0 1 0 1 1

ZOTIMO= 155

NUMERO DE ITERACOES = 11

TEMPO DE CPU UTILIZADO = 0.0033331 SEGUNDOS

*** PROBLEMA 5 ***

J	1	2	3	4	5
MAX Z=	76	67	41	32	10
S.A.	76	67	41	32	10

RECURSO DADO 113
RECURSO UTILIZADO 109

X(J)= 0 1 0 1 1

ZOTIMO= 109

NUMERO DE ITERACOES = 10

TEMPO DE CPU UTILIZADO = 0.0 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	94	93	92	83	82	69	36	16	15	7
S.A.	94	93	92	83	82	69	36	16	15	7

RECURSO DADO 293
RECURSO UTILIZADO 293

X(J)=	0	0	0	1	1	1	1	1	0	1

ZOTIMO= 293

NUMERO DE ITERACOES = 121

TEMPO DE CPU UTILIZADO = 0.0166656 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	91	81	78	74	66	65	40	37	18	15
S.A.	91	81	78	74	66	65	40	37	18	15

RECURSO DADO 282
RECURSO UTILIZADO 282

X(J)=	0	0	0	1	1	1	1	1	0	0

ZOTIMO= 282

NUMERO DE ITERACOES = 165

TEMPO DE CPU UTILIZADO = 0.0233318 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	95	76	67	44	41	36	32	29	29	10
S.A.	95	76	67	44	41	36	32	29	29	10

RECURSO DADO 229
RECURSO UTILIZADO 229

XIJ)=	0	1	0	1	1	0	0	1	1	I

ZOTIMC= 229

NUMERO DE ITERACOES = 173

TEMPO DE CPU UTILIZADO = 0.0199987 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	949	932	921	837	826	816	697	660	650	366	181	167	155	153	72
S.A.	949	932	921	837	826	816	697	660	650	366	181	167	155	153	72

RECURSO DADO 4191
RECURSO UTILIZADO 4191

X(J)=	0	0	0	1	1	0	1	1	1	1	0	0	1	0	0

ZOTIMO= 4191

NUMERO DE ITERACOES = 2250

TEMPO DE CPU UTILIZADO = 0.2633165 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	957	918	781	765	748	673	440	413	404	379	365	326	295	290	107
S.A.	957	918	781	765	748	673	440	413	404	379	365	326	295	290	107

RECURSO DADO 3930
RECURSO UTILIZADO 3930

X(J)=	0	0	1	0	1	1	0	1	1	0	0	1	1	1	0

ZOTIMO= 3930

NUMERO DE ITERACOES = 4629

TEMPO DE CPU UTILIZADO = 0.4899685 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	961	919	883	710	679	657	646	552	440	415	353	323	212	99	14
S.A.	961	919	883	710	679	657	646	552	440	415	353	323	212	99	14

RECURSO DADO 3931
RECURSO UTILIZADO 3931

X(J)=	0	0	1	0	1	1	1	1	0	1	0	0	0	1	0

ZOTIMO= 3931

NUMERO DE ITERACOES = 2258

TEMPO DE CPU UTILIZADO = 0.2966476 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													
MAX Z=	949	932	921	918	837	826	816	781	748	697	660	650	404	379	365	181
167	155	153	72													
S.A.	949	932	921	918	837	826	816	781	748	697	660	650	404	379	365	181
167	155	153	72													

RECURSO DADO 5806
RECURSO UTILIZADO 5806

X(J)=	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	0
0	1	0	1													

ZOTIMO= 5806

NUMERO DE ITERACOES = 67953

TEMPO DE CPU UTILIZADO = 7.9228249 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													
MAX Z=	961	957	919	883	765	710	673	646	552	440	413	365	353	326	323	295
290	212	107	99													
S.A.	961	957	919	883	765	710	673	646	552	440	413	365	353	326	323	295
290	212	107	99													

RECURSO DADO 5144
RECURSO UTILIZADO 5144

X(J)=	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0
1	1	1	1													

ZOTIMO= 5144

NUMERO DE ITERACOES = 86689

TEMPO DE CPU UTILIZADO = 9.8060379 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													
MAX Z=	987	901	868	814	794	793	679	657	578	517	514	440	415	230	156	155
88	67	14	4													
S.A.	987	901	868	814	794	793	679	657	578	517	514	440	415	230	156	155
88	67	14	4													

RECURSO DADO 4810
RECURSO UTILIZADO 4810

X(J)=	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	0
1	0	1	0													

ZOTIMO= 4810

NUMERO DE ITERACOES = 18709

TEMPO DE CPU UTILIZADO = 2.9498100 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25								
MAX Z=	949	932	921	918	837	826	816	781	765	748	697	673	660	650	413	404
379	366	326	181	167	155	153	107	72								
S.A.	949	932	921	918	837	826	816	781	765	748	697	673	660	650	413	404
379	366	326	181	167	155	153	107	72								

RECURSO DADO 6948
RECURSO UTILIZADO 6948

X(J)=	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1
1	1	1	0	1	0	1	1	1								

ZOTIMO= 6948

NUMERO DE ITERACOES = ******

TEMPO DE CPU UTILIZADO = 189.734497 SEGUNDOS

*** PROBLEMA 2 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25								
MAX Z=	961	957	919	883	814	793	710	679	657	646	578	552	440	440	415	365
353	323	295	290	212	105	99	14	4								
S.A.	961	957	919	883	814	793	710	679	657	646	578	552	440	440	415	365
353	323	295	290	212	105	99	14	4								

RECURSO DADO 6252
RECURSO UTILIZADO 6252

X(J)=	0	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1
1	1	1	1	0	1	0	0	0								

ZOTIMO= 6252

NUMERO DE ITERACOES = 723658

TEMPO DE CPU UTILIZADO = 109.166321 SEGUNDOS

*** PROBLEMA 3 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25								
MAX Z=	987	901	868	867	849	802	794	720	719	606	578	577	517	514	251	230
218	214	156	104	88	83	67	47	17								
S.A.	987	901	868	867	849	802	794	720	719	606	578	577	517	514	251	230
218	214	156	104	88	83	67	47	17								

RECURSO DADO 5887
RECURSO UTILIZADO 5887

X(J)=	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	0	1								

ZOTIMO= 5887

NUMERO DE ITERACOES = 750561

TEMPO DE CPU UTILIZADO = 91.2108154 SEGUNDOS

C) PROBLEMA 2 - ALGORITMO ZOLTNERS MODIFICADO

*** PROBLEMA 1 ***

J	1	2	3	4	5
MAX Z=	31	32	33	34	35
S.A.	31	32	33	34	35

RECURSO DADO 70
RECURSO UTILIZADO 69

X(J)= 0 0 0 1 1

ZOTIMO= 69

NUMERO DE ITERACOES = 10

TEMPO DE CPU UTILIZADO = 0.0099994 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	111	112	113	114	115	116	117	118	119	120
S.A.	111	112	113	114	115	116	117	118	119	120

RECURSO DADO 540
RECURSO UTILIZADO 474

X(J)=	0	0	0	0	0	0	1	1	1	1

ZOTIMO= 474

NUMERO DE ITERACOES = 210

TEMPO DE CPU UTILIZADO = 0.1666560 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
S.A.	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

RECURSO DADO 1785
RECURSO UTILIZADO 1764

X(J)=	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

ZOTIMO= 1764

NUMERO DE ITERACOES = 6435

TEMPO DE CPU UTILIZADO = 5.7562942 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													

MAX Z=	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436
437	438	439	440													

S.A.	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436
437	438	439	440													

RECURSO DADO 4160
RECURSO UTILIZADO 3924

X(J)=	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
1	1	1	1													

ZOTIMO= 3924

NUMERO DE ITERACOES = 167960

TEMPO DE CPU UTILIZADO = 195.027496 SEGUNDOS

D) PROBLEMA 2 - ALGORITMO LEXMOCH

*** PROBLEMA 1 ***

J	1	2	3	4	5
MAX Z=	35	34	33	32	31
S.A.	35	34	33	32	31

RECURSO DADO 70
RECURSO UTILIZADO 69

X(J)= 1 1 0 0 0

ZOTIMO= 69

NUMERO DE ITERACOES = 20

TEMPO DE CPU UTILIZADO = 0.0033331 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10
MAX Z=	120	119	118	117	116	115	114	113	112	111
S.A.	120	119	118	117	116	115	114	113	112	111

RECURSO DADO 540
RECURSO UTILIZADO 474

X(J)=	1	1	1	1	0	0	0	0	0	0

ZOTIMO= 474

NUMERO DE ITERACOES = 462

TEMPO DE CPU UTILIZADO = 0.0399974 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MAX Z=	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241
S.A.	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241

RECURSO DADO 1765
RECURSO UTILIZADO 1764

X(J) =	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

ZOTIMO= 1764

NUMERO DE ITERACOES = 12870

TEMPO DE CPU UTILIZADO = 1.0699310 SEGUNDOS

*** PROBLEMA 1 ***

J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20													

MAX Z=	440	439	438	437	436	435	434	433	432	431	430	429	428	427	426	425
424	423	422	421													

S.A.	440	439	438	437	436	435	434	433	432	431	430	429	428	427	426	425
424	423	422	421													

RECURSO DADO 4180
RECURSO UTILIZADO 3924

X(J)=	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0													

ZOTIMO= 3924

NUMERO DE ITERACOES = 352716

TEMPO DE CPU UTILIZADO = 31.3179779 SEGUNDOS

*** PROBLEMA 1 ***

```
   J          1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
  17   18   19   20   21   22   23   24   25

MAX Z=      675  674  673  672  671  670  669  668  667  666  665  664  663  662  661  660
659  658  657  656  655  654  653  652  651

S.A.        675  674  673  672  671  670  669  668  667  666  665  664  663  662  661  660
659  658  657  656  655  654  653  652  651

RECURSO DADO              8100
RECURSO UTILIZADO         8034

X(J)=         1    1    1    1    1    1    1    1    1    1    1    1    0    0    0    0
  0    0    0    0    0    0    0    0    0

ZOTIMO=       8034

NUMERO DE ITERACOES =              ******

TEMPO DE CPU UTILIZADO =     904.888428    SEGUNDOS
```